



# **Informatik III**

## **Grundlagen der theoretischen Informatik**

**Peter Sanders**

**Übungen:**

**Thomas Käufel**

**Roman Dementiev und Dominik Schultes**

Institut für theoretische Informatik, Algorithmik II



# Organisatorisches

**Vorlesungen:** Dienstags und Donnerstags 11.30–13 Uhr

**Saalübung:** Freitags 11.30–13 Uhr, Audimax, Beginn 4. Nov.

**Tutorübungen:** Webinscribe,

`www.ira.uka.de/~thgries/wis/`

**Übungsblätter:**

- Ausgabe: Veröffentlichung auf den Webseiten der Vorlesung – Donnerstags ab 14 Uhr
- Abgabe: Eine Woche später – Freitags, bis 10 Uhr
- Ort: Einwurfkasten beim Raum –118, Informatikgebäude



# Organisatorisches

## Sprechstunde:

- Peter Sanders, Dienstag 14–15 Uhr, Raum 217
- Dr. Th. Käufl, Montag 11–12 Uhr, Raum 207

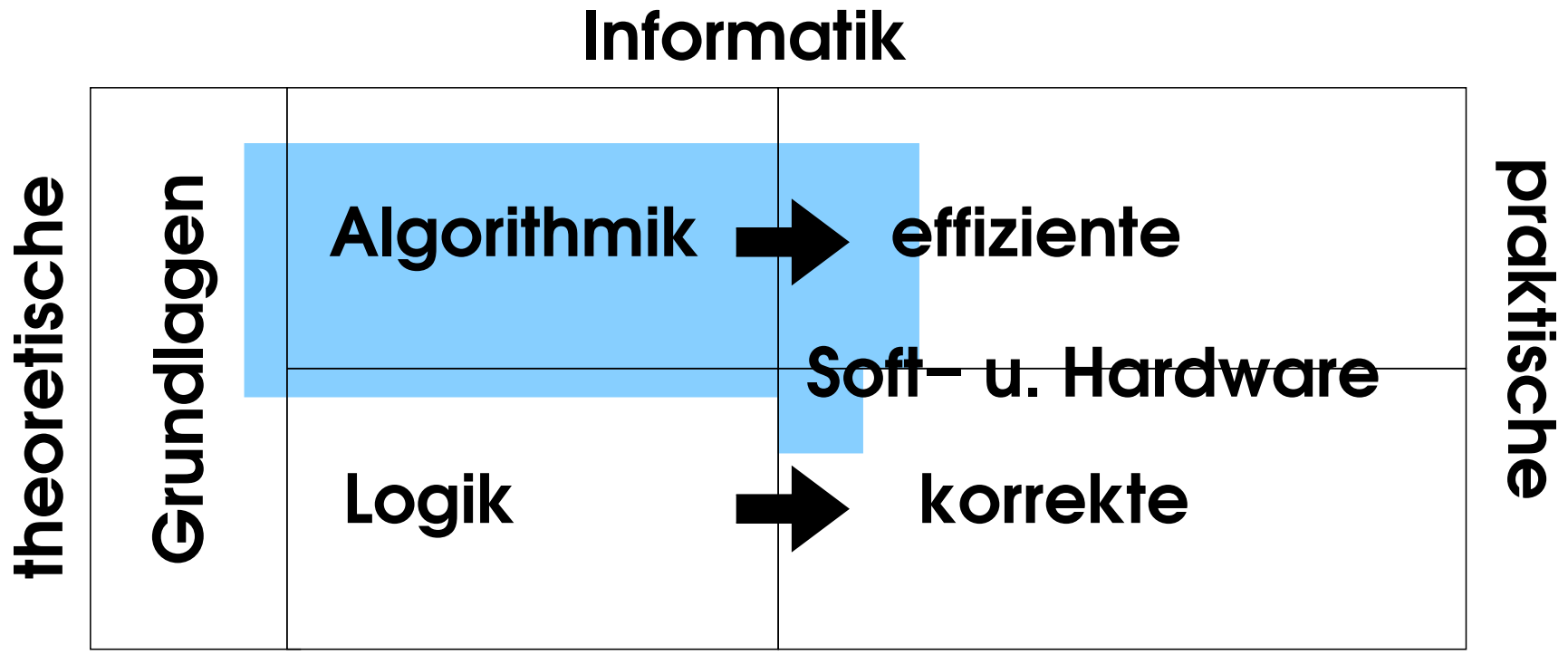
**Klausuren:** Mi 8.3.2006 9:00–11:00 (Wh: Do 20.4.2006 9:00–11:00)

**Web:** [i10www.ira.uka.de/info3/](http://i10www.ira.uka.de/info3/)

Alle Angaben auch auf den Webseiten der Vorlesung.



# Informatik





# 1 Einführung

## Grundlagen theoretischer Informatik

**Formale Sprachen:** Wie kommunizieren Computer?

**Automatentheorie:** Formale Modelle von Computern

**Berechenbarkeit:** Was können Computer (nicht)

**Komplexitätstheorie:** Was kann man (nicht) effizient berechnen



## Warum das Ganze?

- (in)direkte **Anwendungen**  
(Algorithmen, Konzepte, Methoden)  
in Programmiersprachenentwurf, Compilerbau, Textverarbeitung,  
(Verarbeitung natürlicher Sprache), Hardwareentwurf,  
Softwaremodellierung, . . .
- Unmöglichkeitsresultate** ersparen uns blutige Nasen
- Übung mit informatikrelevanten **Beweistechniken**  
↔ Algorithmentheorie, Logik, . . .
- Fester Boden auf dem Grund **philosophischer** Diskussionssümpfe:  
Leib-Seele-Problem, KI, Turing-Test, . . .



# Materialien

## Folien, Übungsblätter

## Bücher: z.B. Schöning oder Wegener

## Skript

- Minimalistisch. Basierend auf Skript Wagner.
- Näher an der Vorlesung
- Wenig Motivation, Anwendungen, Bilder
  - ↪ kein **Ersatz** für die Vorlesung

## Wikipedia!



# Bücher

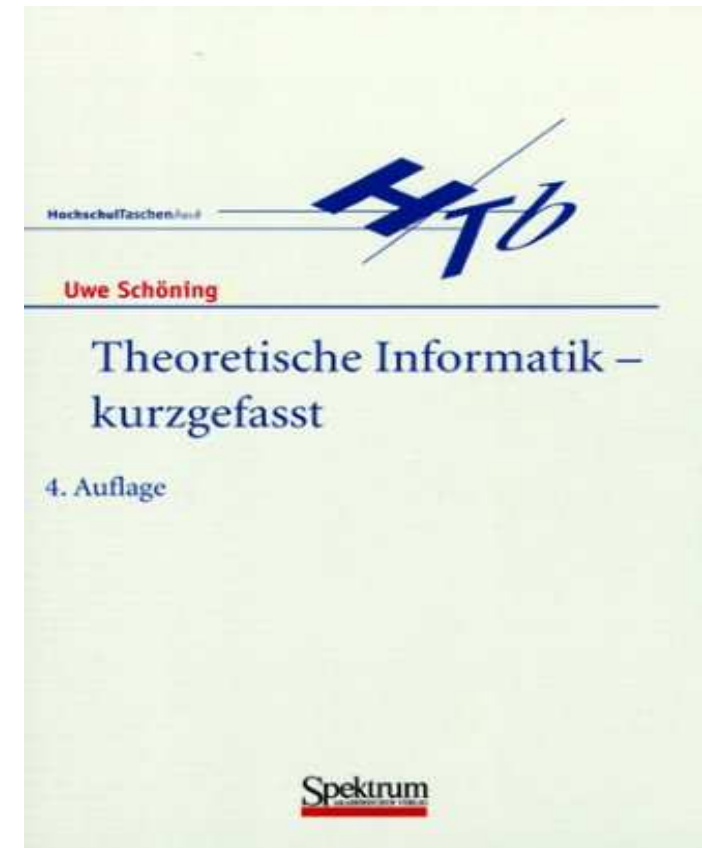
- Wie Sand am Meer
- Vergleichbare Inhalte
- Gut zum Nachlesen und für zusätzliche Motivationen



# Schöning: Theoretische Informatik

## — kurzgefaßt

- 4. Auflage, Spektrum Verlag, 2001, 20 Euro

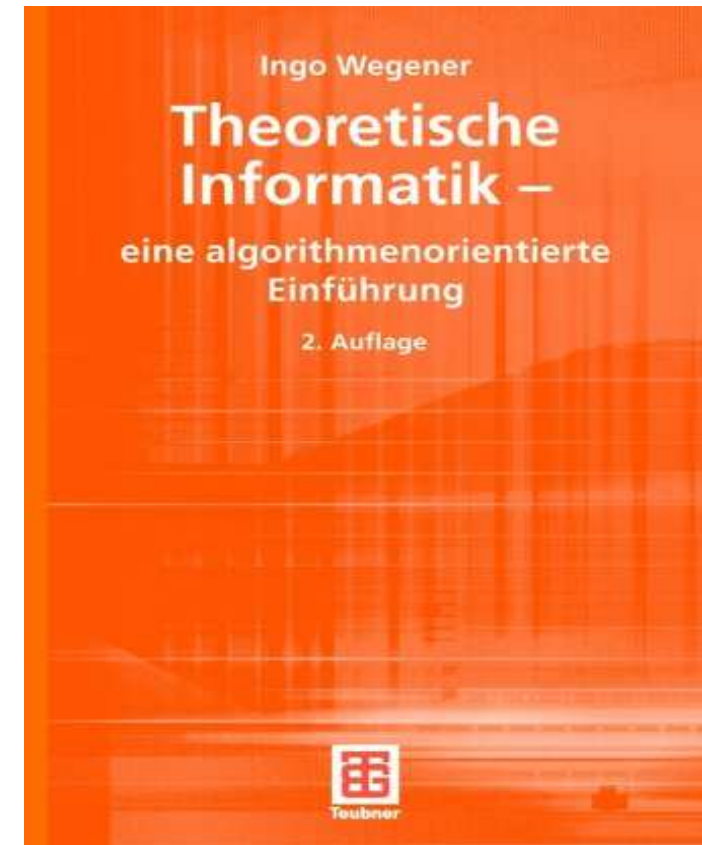




# Wegener: Theoretische Informatik

## — eine algorithmenorientierte Einführung

- 2. Auflage, Teubner, 1999, 22 Euro
- Gute Anwendungsmotivationen
- Zusätzliche, informellere Darstellung:  
Kompendium Theoretische Informatik

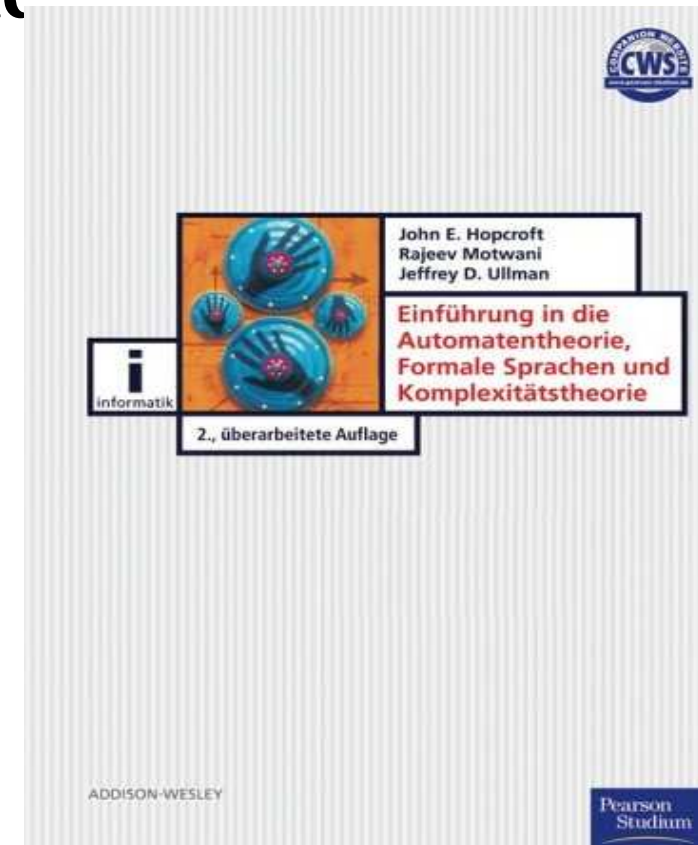




# Hopcroft, Motwani, Ullman

## Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie

- 2. Auflage, Addison-Wesley, 2002, 40 Euro
  - Gute Anwendungsmotivationen
  - Wenig Stoff sehr ausführlich erklärt
  - Geeignet zum Selbststudium?
- ≠ altes Hopcroft Ullman Buch





# 1.1 Formale Sprachen

## Notation

Alphabet: **endliche** Menge ( $\Sigma$ )

Wort (über  $\Sigma$ ): aneinanderghängte Zeichen aus  $\Sigma$  ( $w$ )

formale Sprache: eine Menge von Wörtern ( $L$ )

leeres Wort:  $\epsilon$  ( $\{\epsilon\} \neq \emptyset$  !)

Konkatenation von Zeichenketten:  $\cdot$  assoziativ.

Beispiel:  $ac \cdot bab = acbab$ .

Produktsprache:  $L_1 \cdot L_2 := \{w_1 \cdot w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}$ .

Beispiel:  $\{ab, ba\} \cdot \{aa, bb\} = \{abaa, abbb, baaa, babb\}$



## Notation

$k$ -faches Produkt (Potenzierung):  $w^0 := \varepsilon$ ,  $w^n := w \cdot w^{n-1}$  für  $n \geq 1$ .

$L^0 := \{\varepsilon\}$ ,  $L^n := L \cdot L^{n-1}$  für  $n \geq 1$ .

Beispiele  $a^3 = aaa$ ,  $\{a, bb\}^2 = \{aa, abb, bba, bbbb\}$

Kleensche Hülle:  $L^* := \bigcup_{i=0}^{\infty} L^i$

(jedes einzelne Wort ist immer noch endlich!)

Beispiel:  $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

$\Sigma^*$ : Menge der Wörter über dem Alphabet  $\Sigma$ .

$(\Sigma^*, \cdot, \varepsilon)$  ist ein **Monoid**.

positive Hülle:  $L^+ := \bigcup_{i=1}^{\infty} L^i$

Komplementsprache:  $L^c := \Sigma^* \setminus L$

$|w|$ : Anzahl Buchstaben in  $w$



# Notation

Sei  $w = u \cdot v \cdot x$

Präfix:  $u$

Teilwort:  $v$

Suffix:  $x$

Spiegelung:  $(c_1c_2 \cdots c_k)^R = c_k \cdots c_2c_1$



## Beispiele für formale Sprachen

- $L^= := \{0^n 1^n : n \geq 1\}$
- $\{a^n b^n c^n : n \in \mathbb{N}\}$
- $\{ww : w \in \Sigma^*\}$
- $\{ww^R : w \in \Sigma^*\}$
- $L_P := \{w : w = w^R\}$  **Palindrome**, z.B. anna, otto, aua, gnudung, hangnah, kajak, lagerregal, reliefpfeiler, Saippuakauppias (finnisch Seifenhändler), Sator Arepo Tenet Opera Rotas (Lateinisch; Übersetzung: Der Sähmann Arepo dreht mit Mühe die Räder), tragart „ein Neger mit Gazelle zagt im Regen nie“, „eine Hure bei Liebe ruhe nie“



## Beispiele für formale Sprachen

Wohlgeformte Klammerausdrücke  $L_{()}:$

- $\varepsilon \in L_{()},$
- $uv \in L_{()} \text{ falls } u \in L_{()}, v \in L_{()},$
- $(u) \in L_{()} \text{ falls } u \in L_{()}.$



# Typische Fragestellungen

Wortproblem:  $w \in L$ ?

Äquivalenz:  $L_1 = L_2$ ?

Spezifikation: mathematische Definition (siehe oben), **Grammatiken**,

**Akzeptoren**=Automaten (Maschinen), die Wörter  $w$  lesen und

‘sagen’ ob  $w \in L$

**Umwandlung** zwischen verschiedenen Spezifikationen



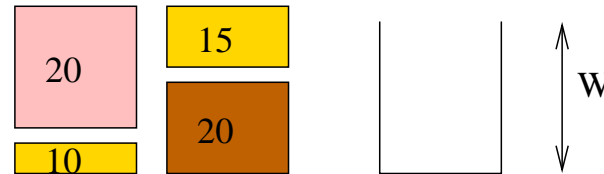
## Warum formale Sprachen?

- Programmiersprachen, Dateiformate, . . . sind formale Sprachen.
- Eingaben im weitesten Sinne (z.B. Impulsfolgen an Pins von Chips) lassen sich als formale Sprachen auffassen
- Einfache klar definierte Fragestellungen
- Die meisten algorithmischen Fragestellung lassen sich auf Wortprobleme zurückführen

„wenn wir formale Sprachen verstehen, verstehen wir die Informatik“



# Beispiel Rucksackproblem



- $n$  Gegenstände mit **Gewicht**  $w_i \in \mathbb{N}$  und **profit**  $p_i$
- Wähle eine Teilmenge **x** von Gegenständen
- so dass  $\sum_{i \in \mathbf{x}} w_i \leq W$  und
- maximiere den Profit**  $\sum_{i \in \mathbf{x}} p_i$



## Beispiel Rucksackproblem

definiere  $L \subseteq \{0, 1, ','\}^*$ :

$w \in L$  falls  $w$  eine kommaseparierte Liste von  $2n + 2$  Binärzahlen

$P, W, w_1, p_1, \dots, w_n, p_n$  ist, so dass

$$\exists \mathbf{x} \subseteq \{1, \dots, n\} : \sum_{i \in \mathbf{x}} p_i \geq P \text{ und } \sum_{i \in \mathbf{x}} w_i \leq W$$

**Wortproblem für  $L \rightsquigarrow$  Rucksackproblem:**

- rate optimales  $P$  durch **binäre Suche**
- finde  $\mathbf{x}$  durch weglassen einzelner Elemente

Insgesamt  $\leq n + \log \sum_i p_i$  Wortprobleme lösen

„wenig“

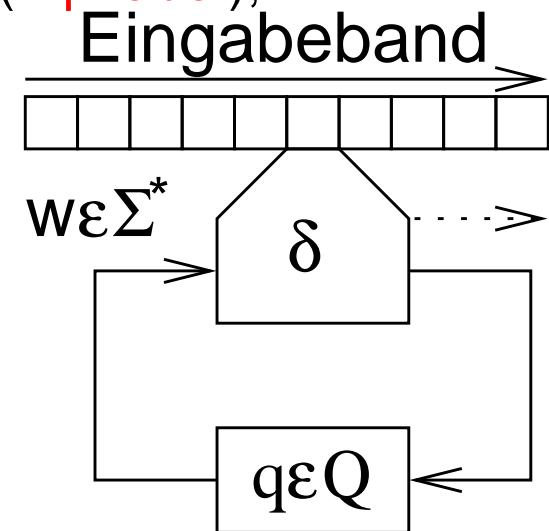


## 1.2 Automatentheorie

### Endliche Automaten

Ein deterministischer endlicher Automat (oder Akzeptor) besteht aus:

- $Q$ , einer endlichen Menge von **Zuständen**;
- $\Sigma$ , einer endlichen Menge von **Eingabesymbolen**, (**Alphabet**);
- $\delta: Q \times \Sigma \rightarrow Q$ , einer **Übergangsfunktion**;
- $s \in Q$ , einem **Startzustand**;
- $F \subseteq Q$ , einer Menge von **Endzuständen**.





## Was tut ein endlicher Automat?

Wir erweitern die Def. von  $\delta$  für Wörter:

$$\delta(q, \varepsilon) = q$$

$$\delta(q, wa) = \delta(\delta(q, w), a)$$

— ein Zustandsübergang pro

Eingabezeichen

$A = (Q, \Sigma, \delta, s, F)$  akzeptiert die Sprache

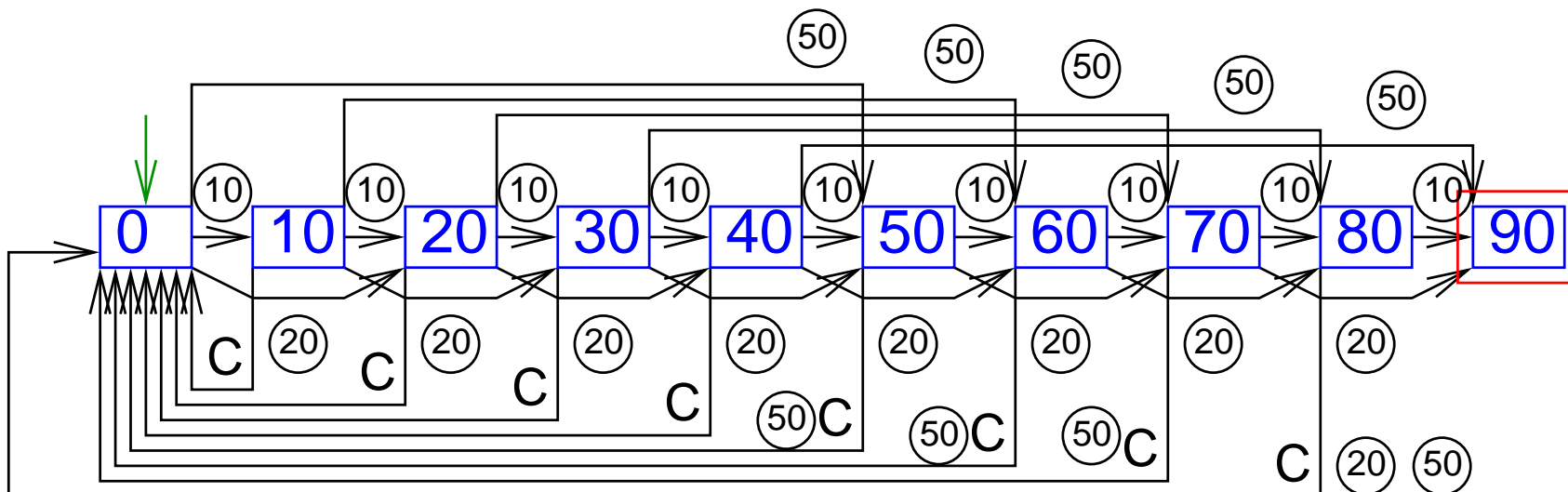
$$L(A) := \{w \in \Sigma^* : \delta(s, w) \in F\}$$



# Beispiel: einfacher Fahrkartenautomat

- Einheitsfahrpreis 90 Cents
- 10, 20 und 50 Centstücke werden akzeptiert
- Abbruch bei Taste C oder zu viel Geld
- Genau 90 Cents eingeworfen  $\rightsquigarrow$  fertig

$(\{10, 20, 50, C\}, \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90\}, \delta, 0, \{90\})$





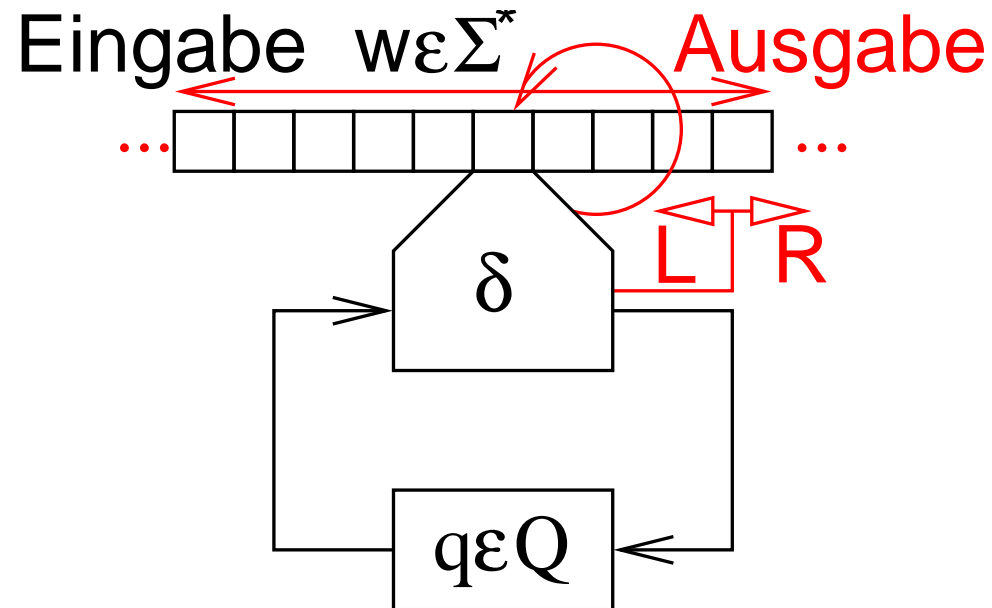
## Die nächsten Vorlesungen

- Welche formalen Sprachen werden durch EA akzeptiert?
- Systematische Konstruktion von EA zu einer Sprache und „umgekehrt“
- Zustandsminimierung und Äquivalenz von EA
- Was können EA **nicht**

„Wir **beherrschen** endliche Automaten vollständig“



# 1.3 Mächtigere Maschinen: Turingmaschinen und **Berechenbarkeit**



- Turingmaschinen können nicht mehr und nicht weniger als alle anderen **hinreichend mächtigen Maschinenmodelle**
- Es gibt (wichtige) **nichtberechenbare Funktionen**. Insbesondere können wir kaum etwas mit beliebigen Turingmaschinen „machen“



## 1.4 Komplexitätstheorie: Was können wir **effizient** berechnen?

- Wieder sind Turingmaschinen (fast) ObdA. Wir vernachlässigen polynomiell große Faktoren in der Laufzeit (als Funktion der Eingabegröße).

$$n \approx n^2 \approx \dots n^{42} \approx \dots \ll 2^{0.134n}$$

- Wir wissen wenig über **untere Laufzeitschranken**
- Aber es gibt große Klassen von wichtigen algorithmischen Problemen, von denen **alle oder keines** effizient lösbar sind



## 1.5 Was gibt es zwischen endlichen Automaten und Turingmaschinen?

Chomsky-Hierarchie: Grammatiken, Automatenmodelle, und passende Familien von formalen Sprachen.

Insbesondere: kontextfreie Sprachen. Zum Beispiel zwecks Spezifikation des Syntax von Programmiersprachen.

- Kontextfreie Sprachen effizient erkennen
- Welche Sprachen können wir in **linearer** Zeit erkennen?



## 2 Endliche Automaten und Reguläre Ausdrücke

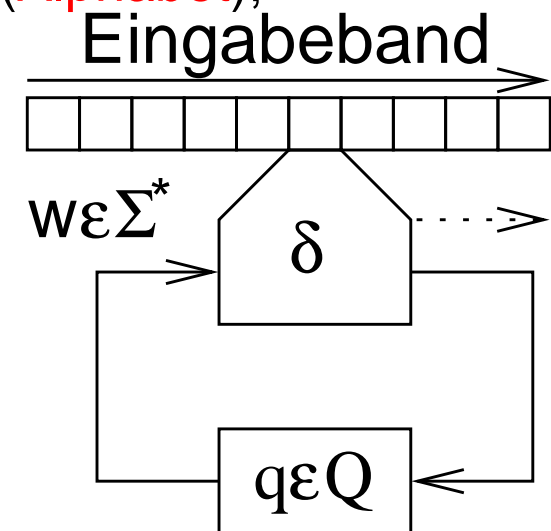
Für welche Sprachen  $L$  gibt es einen endlichen Automaten  $A$  der  $L$  akzeptiert?



# Endliche Automaten

Ein deterministischer endlicher Automat (oder Akzeptor) besteht aus:

- $Q$ , einer endlichen Menge von **Zuständen**;
- $\Sigma$ , einer endlichen Menge von **Eingabesymbolen**, (**Alphabet**);
- $\delta: Q \times \Sigma \rightarrow Q$ , einer **Übergangsfunktion**;
- $s \in Q$ , einem **Startzustand**;
- $F \subseteq Q$ , einer Menge von **Endzuständen**.





# Graphinterpretation

$$A = (Q, \Sigma, \delta, s, F)$$

$$G_A = (Q, E)$$

mit  $e = (q, q') \in E$ , **Beschriftung**  $\ell(e) = a$  falls  $q' = \delta(q, a)$

**Multigraph!**

**Lemma:**

$$\forall w \in \Sigma^* : w \in L(A) \Leftrightarrow$$

$$\exists \text{Abarbeitungspfad } P = sq_1q_2 \cdots f = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} f$$

mit  $f \in F$ , der mit  $w = a_1a_2 \cdots a_k$  beschriftet ist.

**Beweis:** Übung

**Terminologie:**

Wenn wir von Pfaden in  $A$  reden, meinen wir Pfade in  $G_A$ .



## Notation für Pfade

$P = sq_1q_2 \cdots f$  Folge von **Knoten**

$P = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} f$  Folge von (direkten) **Übergängen**

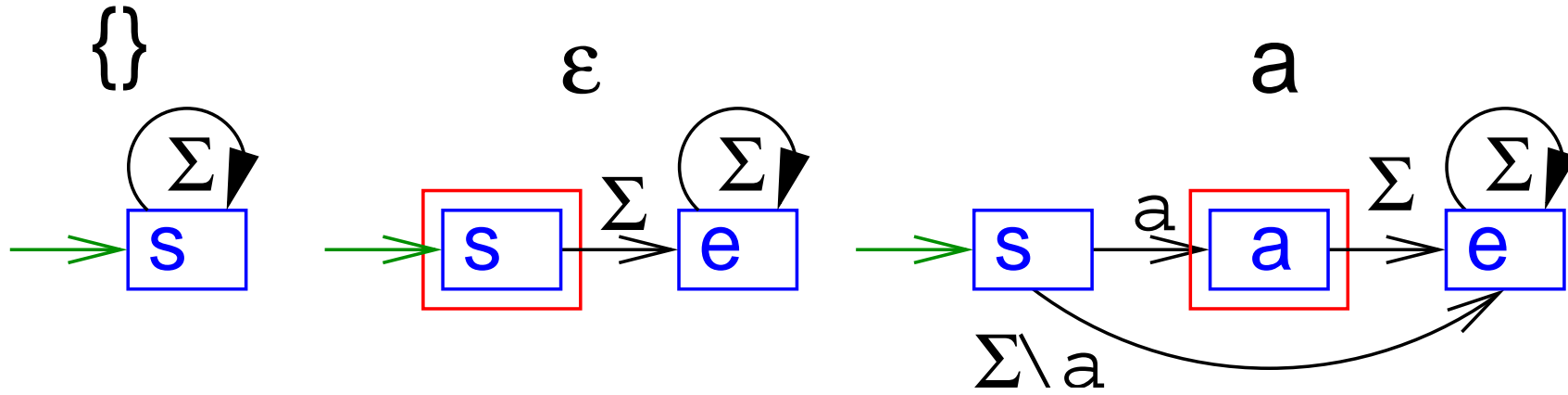
$P = s \xrightarrow{w}$  mit  $w = a_1 \cdots a_k$ .  $P$  ist mit  $w$  **beschriftet**

$q \Rightarrow r$  es gibt Pfad von  $q$  nach  $r$ , d.h.,  $r$  ist von  $q$  aus **erreichbar**

Es gilt  $s \Rightarrow s$  (Reflexivität)

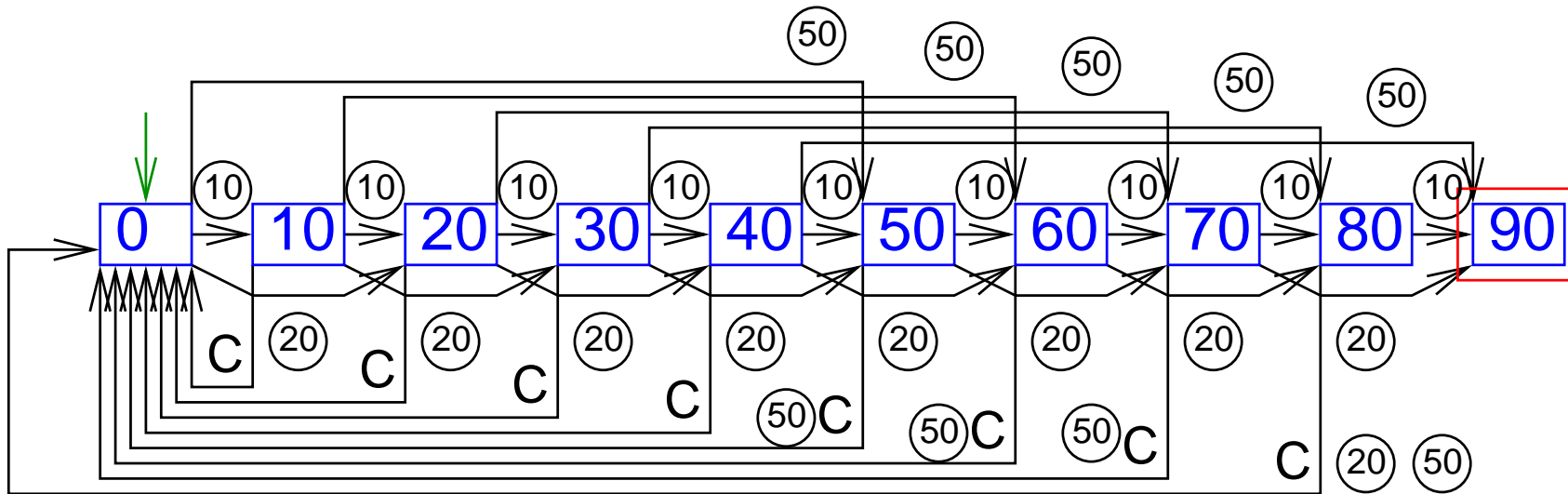


# Einfache Beispiele





# Beispiel: Erfolgreiche Bezahlvorgänge





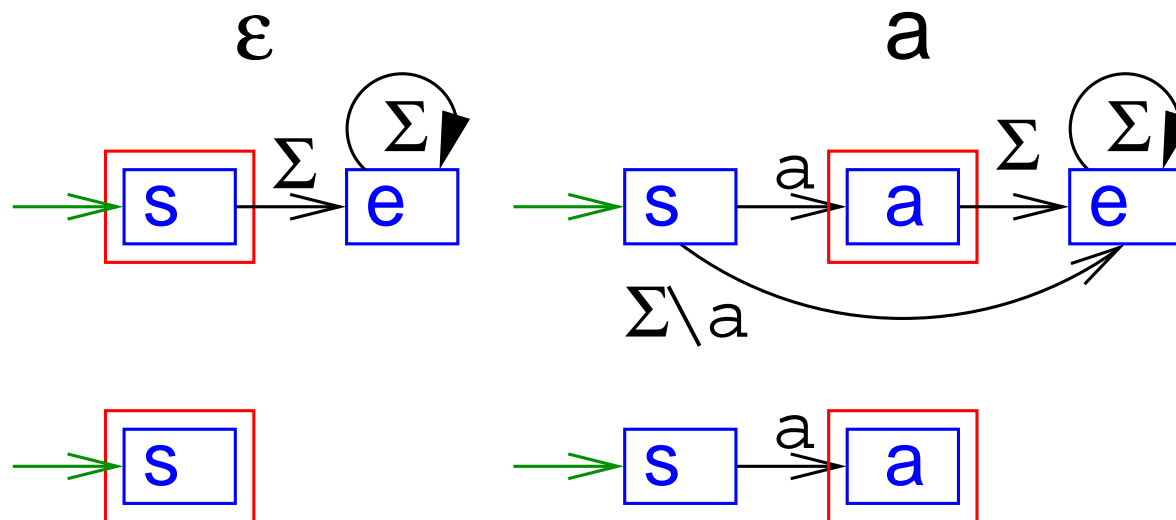
# Vervollständigung

Oft geben wir nicht alle Funktionswerte von  $\delta$  an.

Konvention: Es gibt immer einen Fehlerzustand  $e$  mit

$\delta(q, c) = e$  wenn kein anderer Wert angegeben.

$\delta(e, c) = e \forall c \in \Sigma$





## 2.1 Reguläre Sprachen

Eine Sprache  $L \subseteq \Sigma^*$  heißt **regulär**, wenn : (induktive Definition)

I. Verankerung:

(1)  $L = \{a\}$  mit  $a \in \Sigma$  oder

(2)  $L = \emptyset$

(3)  $L = \{\varepsilon\}$

II. Induktion: Seien  $L_1, L_2$  reguläre Sprachen

(3)  $L = L_1 \cdot L_2$  oder

(4)  $L = L_1 \cup L_2$  oder

(5)  $L = L_1^*$



## Beispiele

- vorletztes Zeichen 0:

$$(\{0\} \cup \{1\})^* \cdot \{0\} \cdot (\{0\} \cup \{1\})$$

- Bezeichner in vielen Programmiersprachen:

$$(\{a\} \cup \dots \cup \{Z\}) \cdot (\{a\} \cup \dots \cup \{Z\} \cup \{0\} \cup \dots \cup \{9\})^*$$



# Reguläre Ausdrücke

Ein reg. Ausdruck **beschreibt** eine reguläre Sprache.

Ausdruck	beschreibt	Bemerkung
$\emptyset$	$\emptyset$	
$\varepsilon$	$\{\varepsilon\}$	
$a$	$\{a\}$	$a \in \Sigma$
$\alpha \cup \beta$	$L(\alpha) \cup L(\beta)$	$\alpha$ beschreibt $L(\alpha)$
$\alpha \cdot \beta$	$L(\alpha) \cdot L(\beta)$	$\beta$ beschreibt $L(\beta)$
$(\alpha)$	$L(\alpha)$	
$\alpha^*$	$L(\alpha)^*$	
$\alpha^+$	$L(\alpha)^+$	

Nachlässigkeiten: ‘.’ weglassen,  $L(\cdot)$  weglassen



# **Syntax** (reg. Ausdrücke) versus **Semantik** (reg. Sprachen)

Computerprogramme bearbeiten **syntaktische** Objekte.

## **Programmverifikation**

Wir beweisen auf **semantischer** Ebene, daß die Objekte korrekt verarbeitet werden.



## Beispiele

- vorletztes Zeichen 0:  $(0 \cup 1)^* 0 (0 \cup 1)$
- enthält 10:  $(0 \cup 1)^* 10 (0 \cup 1)^*$
- enthält nicht 10:  $0^* 1^*$
- enthält 101:  $(0 \cup 1)^* 101 (0 \cup 1)^*$
- enthält nicht 101:  $0^* 1^* \cup (0^* 1^* 100)^* 0^* 1^* 10 (\epsilon \cup 00^* 1^*)$
- ganze Zahl:  $(\epsilon \cup + \cup -) (1 \cup \dots \cup 9) (0 \cup \dots \cup 9)^* \cup 0$



# Erstes Hauptergebnis

Satz: Die **regulären Sprachen** sind identisch mit den Sprachen, die von **endlichen Automaten** akzeptiert werden.

zu zeigen:

→ regulärer Ausdruck  $\alpha \rightsquigarrow$  EA  $A$  mit  $L(\alpha) = L(A)$

← EA  $A \rightsquigarrow$  regulärer Ausdruck  $\alpha$  mit  $L(\alpha) = L(A)$



# Regulärer Ausdruck $\rightarrow$ EA

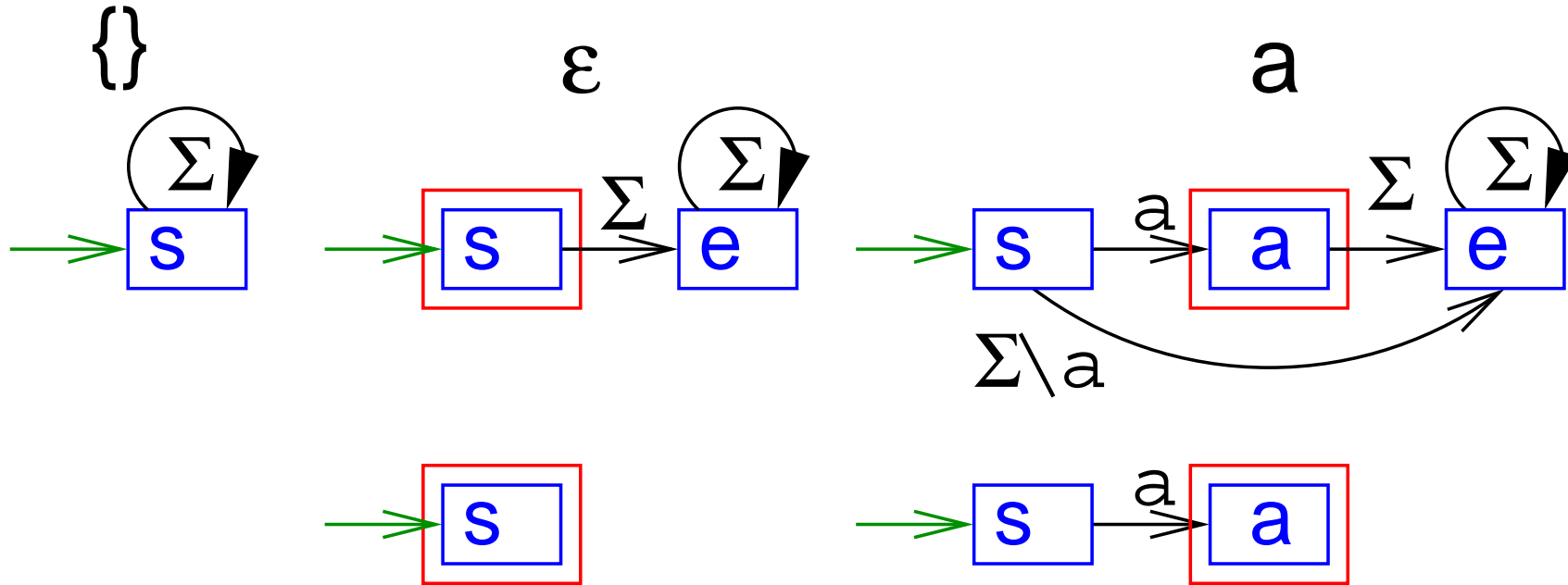
Beweisidee:

baue Automaten für Teilausdrücke

stöpsle diese zu Automaten für komplexere Ausdrücke zusammen



# Basisfälle





# Oops

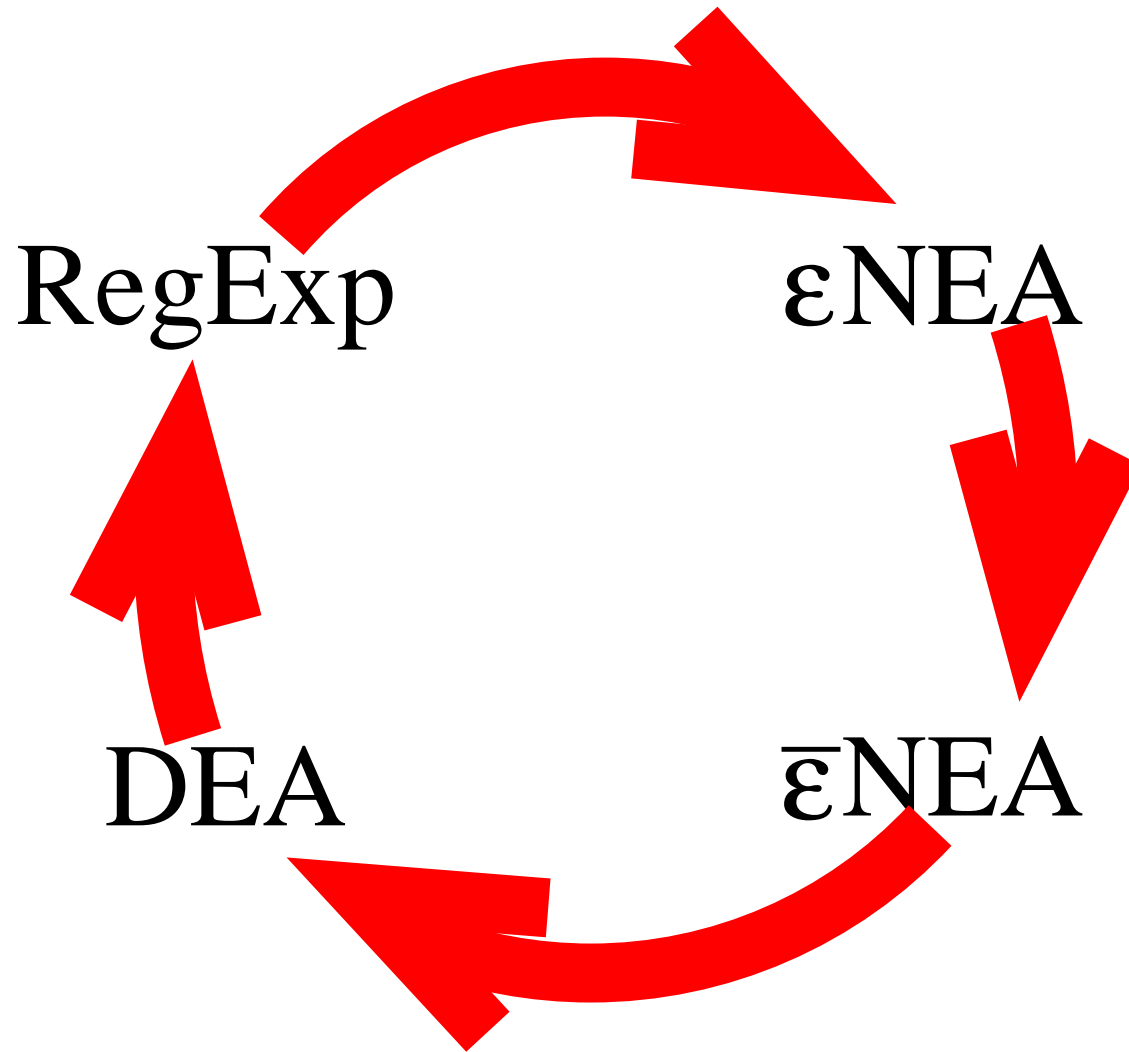
**Problem  $\alpha \cup \beta$ :** was wenn es Wörter in  $L(\alpha)$  und  $L(\beta)$  mit **gemeinsamem** Präfix gibt? Man müßte zu beiden Teilautomaten gleichzeitig verzweigen.

**Problem  $\alpha \cdot \beta$ :** was wenn es Suffixe von Wörtern in  $L(\alpha)$  gibt, die Präfixe von Wörtern in  $L(\beta)$  sind?

**Problem  $\alpha^*$**  wenn in Endzustand zu  $\alpha$ . Weiter mit  $\alpha$  oder von vorn?



# Der Plan





## $\epsilon$ -Übergänge

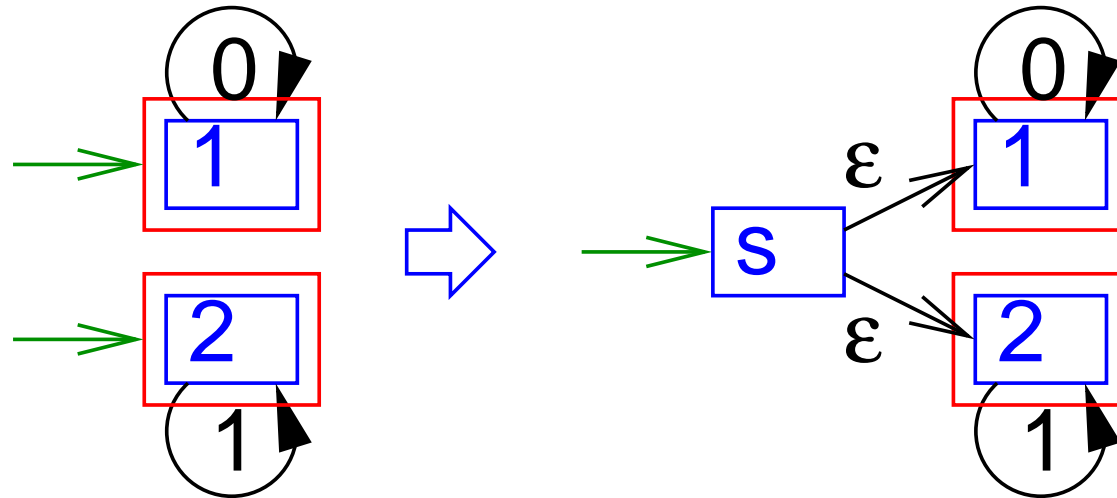
dirty trick:

wir erlauben **spontane** Übergänge

ohne Verbrauch eines Eingabezeichens.



# Beispiel: $0^* \cup 1^*$





## 2.2 **Nichtdeterministische** endliche Automaten

### ( $\epsilon$ )NEA

- erlaube spontane  $\epsilon$ -Übergänge ohne Verbrauch eines Eingabezeichens ( $\epsilon$ NEA sonst  $\bar{\epsilon}$ NEA)
- mehrere erlaubte Übergänge für gegebenes Situation (Zustand, Eingabezeichen)



## Nichtdeterministische endlicher Automat A

- $Q$ , Zustandsmenge
- $\Sigma$ , Eingabealphabet
- $\delta: Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$ , Übergangsfunktion
- $s \in Q$ , Startzustand
- $F \subseteq Q$ , Endzustände

**Spontaner** Zustandsübergang von  $q$  nach  $q'$ :  $q' \in \delta(q, \epsilon)$

Zustandsübergang von  $q$  nach  $q'$  bei Eingabe von  $a$ :  $q' \in \delta(q, a)$

i.allg. mehrere Möglichkeiten!



# Nichtdeterministischer endlicher Automat $A$

Variante (äquivalent) —  $\delta$  als Relation.

- $Q$ , Zustandsmenge
- $\Sigma$ , Eingabealphabet
- $\delta \subseteq Q \times (\Sigma \cup \varepsilon) \times Q$  Übergangsrelation
- $s \in Q$ , Startzustand
- $F \subseteq Q$ , Endzustände

$A$  **kann** Zustandsübergang von  $q$  nach  $q'$  machen wenn  $(q, \varepsilon, q') \in \delta$  oder  $a$  eingegeben wird und  $(q, a, q') \in \delta$ .



## (Multi)Graphinterpretation für $L(A)$

$$A = (Q, \Sigma, \delta, s, F)$$

$$G_A = (Q, E)$$

Funktionsinterpretation von  $\delta$

mit  $e = (q, q') \in E$ , **Beschriftung**  $\ell(e) = a$  falls  $q' \in \delta(q, a)$

$$G_A = (Q, \delta)$$

Relationeninterpretation von  $\delta$

schreibe Kanten  $e$  mit  $\ell(e) = a$  als  $(q, a, q')$ .

„Multi“=parallele Kanten erlaubt (unterschiedliche Beschriftungen)

$$w \in L(A) \Leftrightarrow \exists \text{ Pfad } P = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_k} f \text{ in } A \text{ (in } G(A)) :$$

$$f \in F \wedge w = a_1 a_2 \dots a_k$$

In Worten: Ein Pfad von  $s$  zu einem Endzustand ist mit  $w$  **beschriftet**.



## Beobachtungen zur Definition von $L(A)$

$$A = (Q, \Sigma, \delta, s, F)$$

$$G_A = (Q, \delta)$$

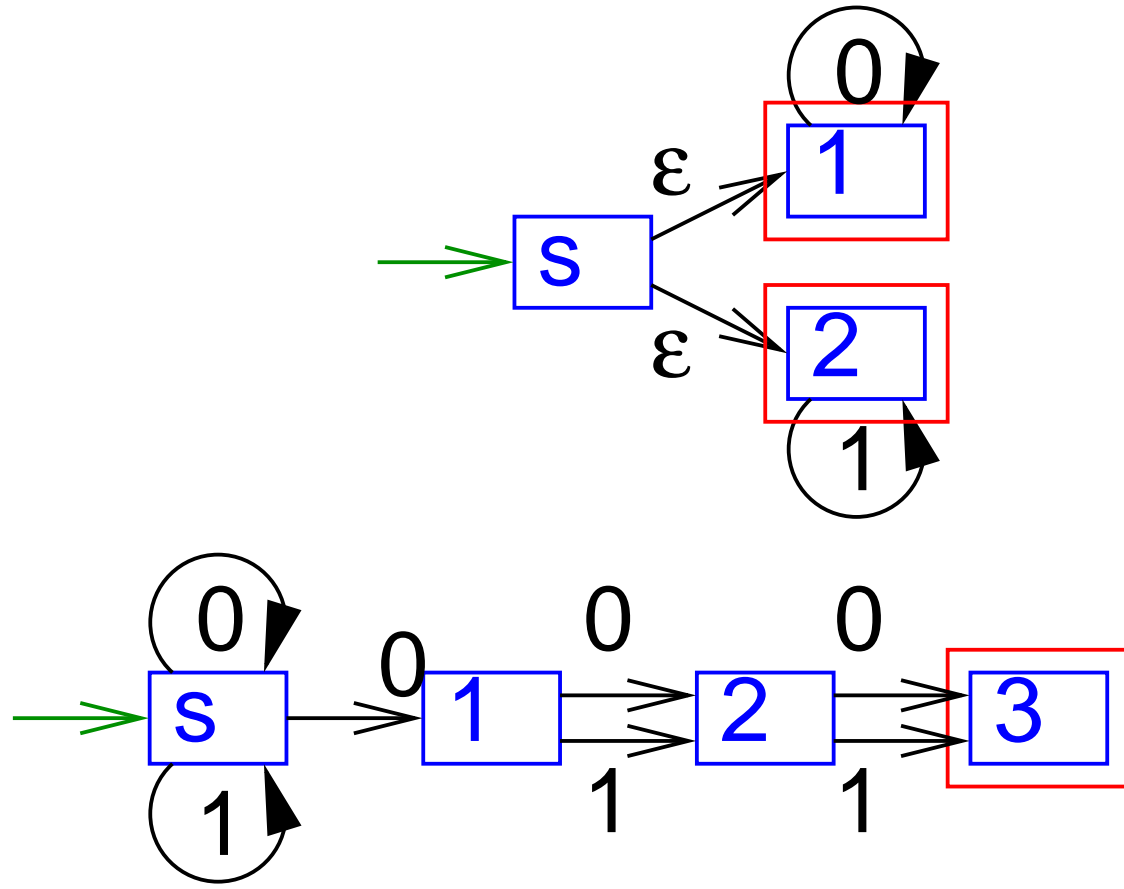
schreibe Kanten  $e$  mit  $\ell(e) = a$  als  $(q, a, q')$ .

$$w \in L(A) \Leftrightarrow \exists \text{ Pfad } P = s \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} f \text{ in } G : \\ f \in F \wedge w = a_1 a_2 \cdots a_k$$

- Es kann viele Pfade für ein  $w$  geben.
- Nicht alle müssen zu einem Endzustand führen!
- $\varepsilon$ -Beschriftungen  $\rightsquigarrow |w| \leq k$ .



# Beispiele





# RegExp $\rightarrow$ $\epsilon$ NEA: $L_1 \cup L_2$

$A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  mit  $L(A_1) = L_1$

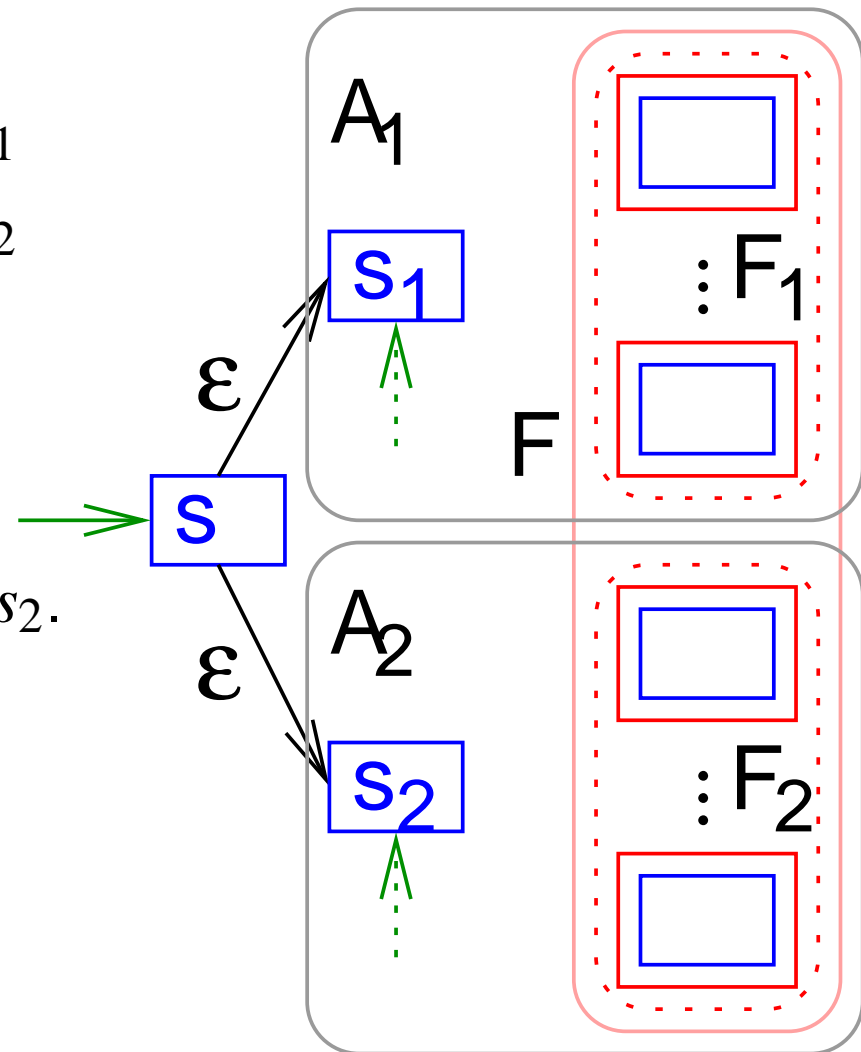
$A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$  mit  $L(A_2) = L_2$

sowie  $Q_1 \cap Q_2 = \emptyset$

$A := (\{s\} \cup Q_1 \cup Q_2, \Sigma, \delta, s, F_1 \cup F_2)$

$\delta$  verhält sich wie  $\delta_{1/2}$  für  $Q_{1/2}$

spontane Übergänge von  $s$  nach  $s_1$  und  $s_2$ .





**Beweis von  $L_1 \cup L_2 \subseteq L(A)$**

$w \in L_1 = L(A_1)$

$\rightarrow \exists$  Pfad  $P = s_1 \rightsquigarrow f_1 \in F_1 \subseteq F$

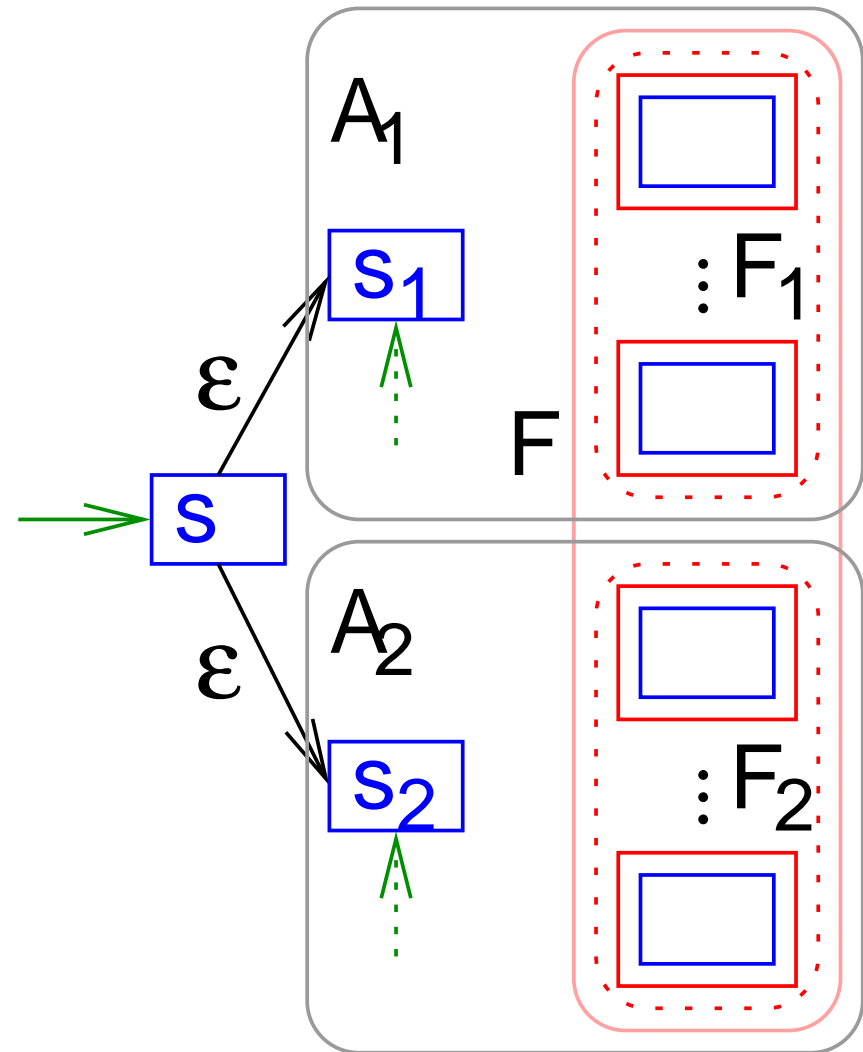
durch  $A_1$  mit Beschriftung  $w$

$\rightarrow sP$  hat Beschriftung  $w$

$\rightarrow w \in L(A)$ .

$w \in L_2 = L(A_2)$

$\rightarrow \dots \rightarrow w \in L(A)$ .





### Beweis von $L(A) \subseteq L_1 \cup L_2$

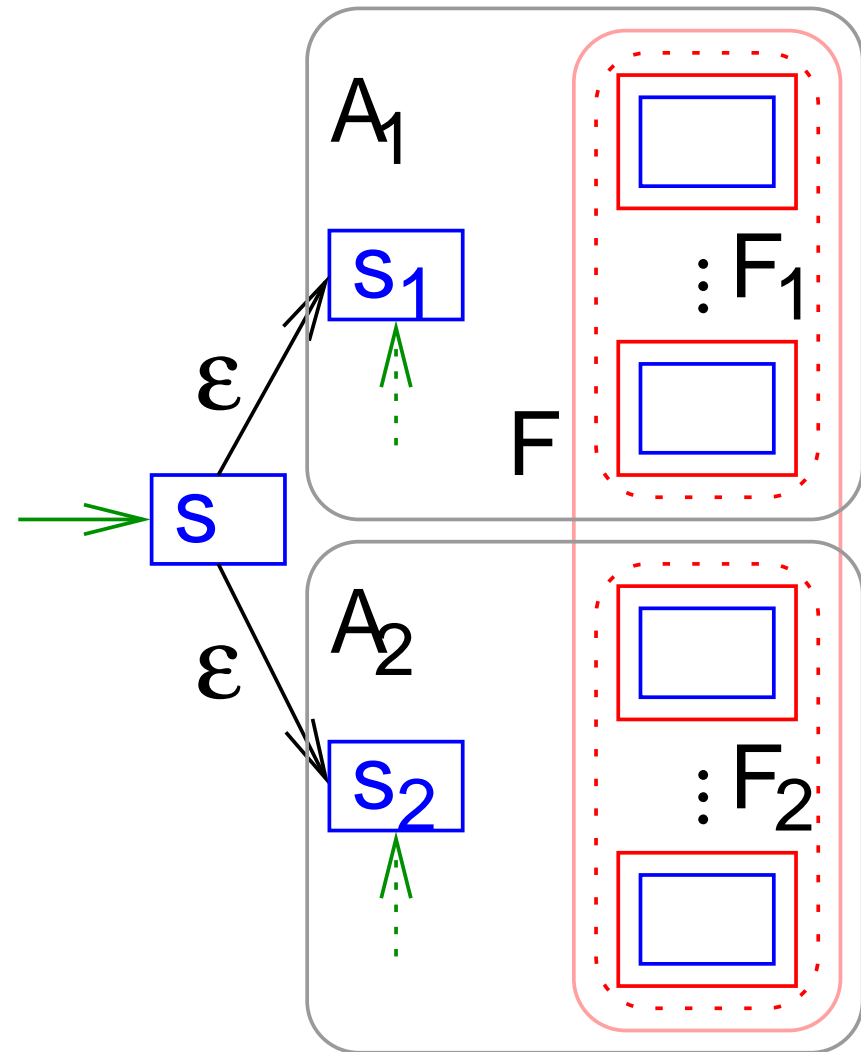
Sei  $w \in L(A)$  beliebig.

$\rightarrow \exists$  Pfad  $P = ss_1 \rightsquigarrow f_1 \in F_1$  oder

$P = ss_2 \rightsquigarrow f_2 \in F_2$

$\rightarrow w \in L(A_1) \vee w \in L(A_2)$

$\leftrightarrow w \in L(A_1) \cup L(A_2)$





$$L_1 \cdot L_2$$

$A_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  mit  $L(A_1) = L_1$

$A_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$  mit  $L(A_2) = L_2$

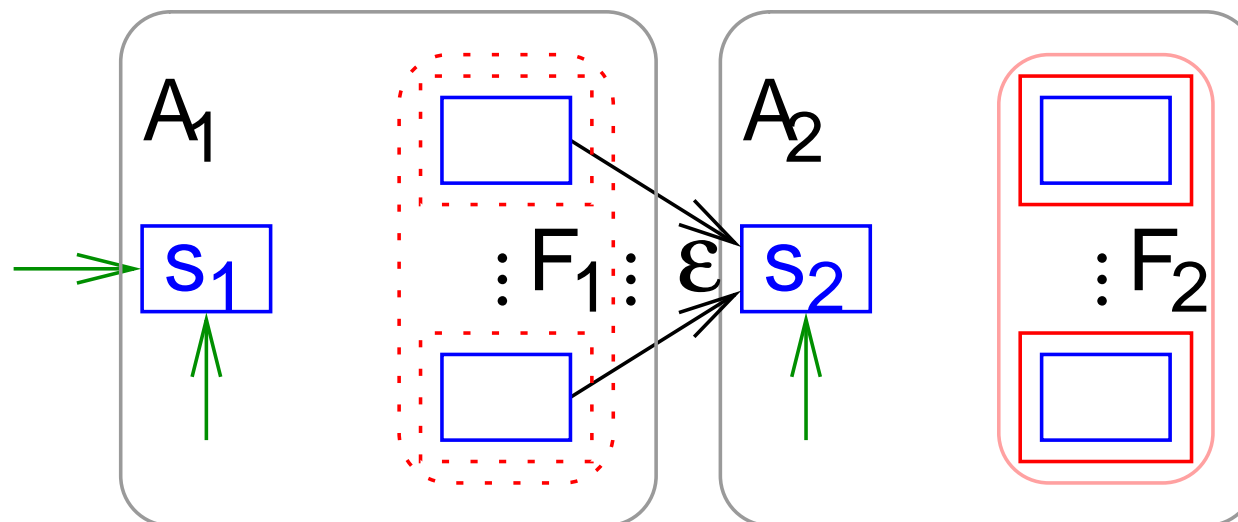
sowie  $Q_1 \cap Q_2 = \emptyset$

$A := (Q_1 \cup Q_2, \Sigma, \delta, s_1, F_2)$

$\delta$  verhält sich wie  $\delta_{1/2}$  für  $Q_{1/2}$

spontane Übergänge von  $F_1$  nach  $s_2$ .

Beweis:...





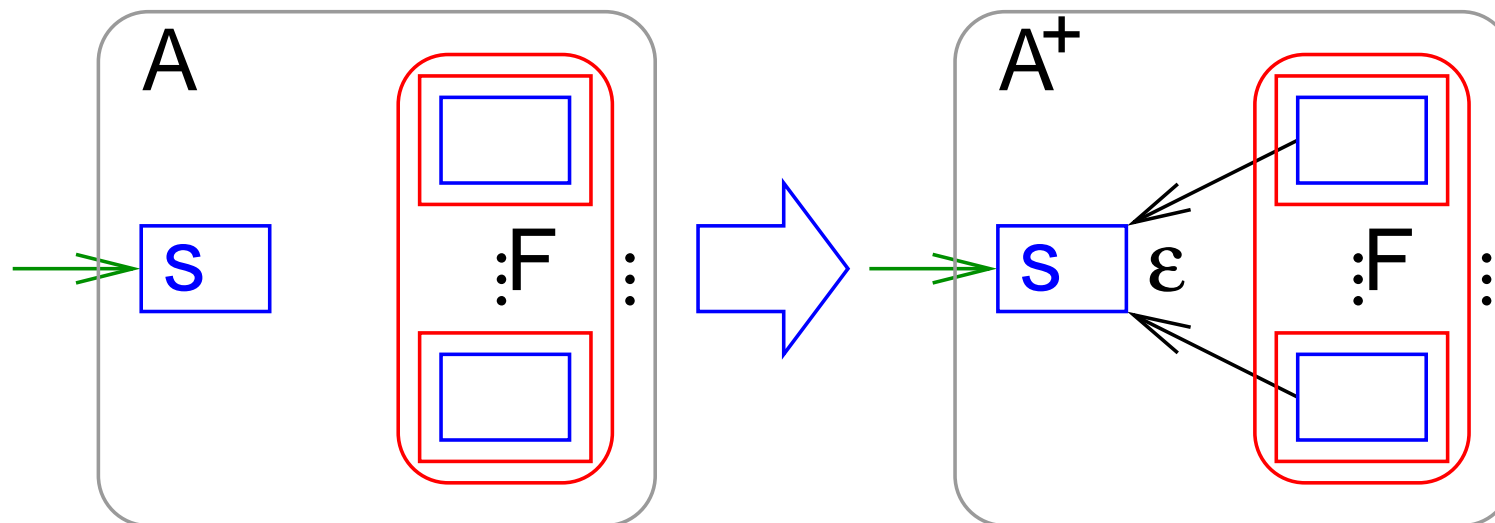
**Positive Hülle**  $L^+ = \bigcup_{i \geq 1} L^i$

$A = (Q, \Sigma, \delta, s, F)$  mit  $L(A) = L$

$A^+ := (Q, \Sigma, \delta^+, s, F)$

$\delta^+$  verhält sich wie  $\delta$

spontane Übergänge  $f \rightarrow s \forall f \in F$ .





## Beweis von $L(A^+) \subseteq L^+$

Sei  $w \in L(A^+)$  beliebig

Sei  $P$  ein akzeptierender Pfad für  $w$ .

Zerlege  $P = P_1 \xrightarrow{\varepsilon} P_2 \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} P_i$

an allen  $f \xrightarrow{\varepsilon} s$ -Übergängen mit  $f \in F$ .

$P_j = s \Rightarrow f \in F$  (Fälle  $j = 1, 1 < j < i, j = i$ ),

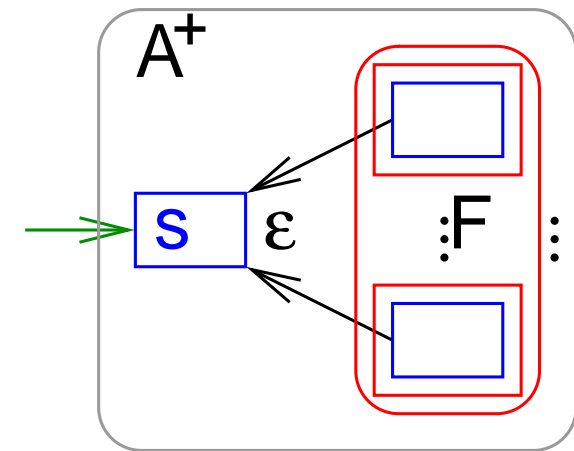
$P_j$  enthält nur Übergänge aus  $\delta$ .

$\rightarrow P_j$  ist akzeptierender Pfad in  $A$

$\rightarrow P_j$  ist mit  $w_j \in L$  beschriftet.

$\rightarrow P$  ist mit  $w = w_1 \varepsilon w_2 \varepsilon \dots \varepsilon w_i$  beschriftet,

$w_1 \varepsilon w_2 \varepsilon \dots \varepsilon w_i = w_1 w_2 \dots w_i \in L^i \subseteq L^+$





# Beweis von $L^i \subseteq L(A^+)$ für $i \geq 1$

Sei  $w = w_1 \cdots w_i \in L^i$  beliebig.

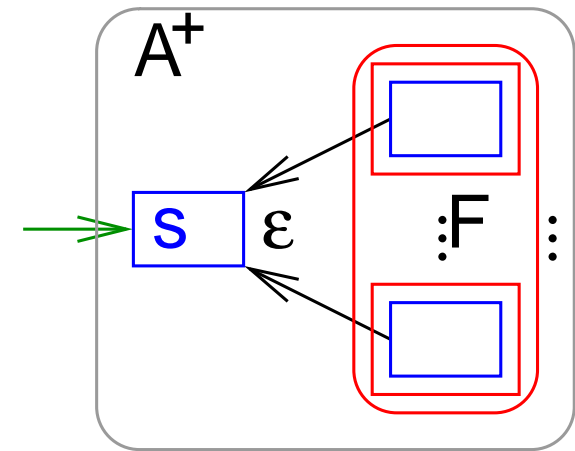
Betrachte Pfade  $P_1, \dots, P_i$  die

$w_1 \in L, \dots, w_i \in L$  bezeugen.

$\rightarrow P = P_1 \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} P_i$  ist ein Pfad in  $A^+$ ,

der  $w \in L(A^+)$  bezeugt.

$\rightarrow w \in L(A^+)$





# Kleensche Hülle $L^*$

Baue Automaten für  $\varepsilon \cup L^+ = L^*$ .



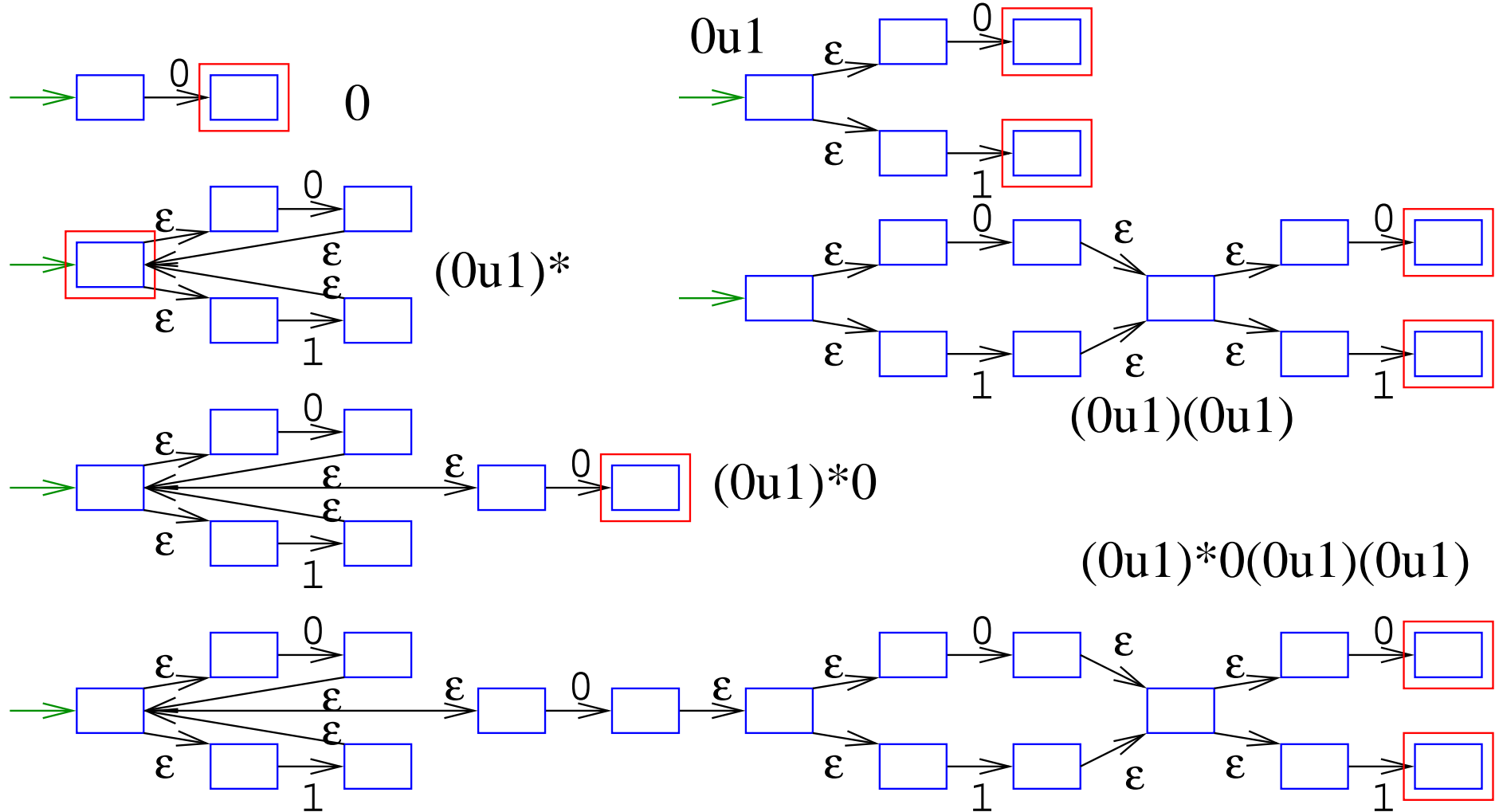


# RegExp $\rightarrow$ $\epsilon$ NEA

Satz: zu jedem regulären Ausdruck  $R$  gibt es einen  $\epsilon$ NEA  $A$  mit  $L(A) = L(R)$ .



# Beispiele





$$\varepsilon\text{NEA} \rightarrow \bar{\varepsilon}\text{NEA}$$

$$E(q) := \{p \in Q : p \text{ ist von } q \text{ durch } \varepsilon\text{-Übergänge } \text{erreichbar}\}$$

Bemerkung:  $\forall q : q \in E(q)$

Eingabe:  $A = (Q, \Sigma, \delta, s, F)$

Ausgabe:  $\tilde{A} := (Q, \Sigma, \tilde{\delta}, s, \tilde{F})$  mit

$$\tilde{\delta} : Q \times \Sigma \rightarrow 2^Q :$$

$$\tilde{\delta}(q, a) := \bigcup_{p \in E(q)} \delta(p, a)$$

„ $\varepsilon^* \Sigma$ “

und

$$\tilde{F} := \{\tilde{f} \in Q : E(\tilde{f}) \cap F \neq \emptyset\}$$

**Satz:**  $L(\tilde{A}) = L(A)$



**Satz:**  $L(\tilde{A}) = L(A)$

**Beweis für  $L(\tilde{A}) \subseteq L(A)$**

Sei  $w \in L(\tilde{A})$  beliebig.

Sei  $\tilde{P} = s \Rightarrow \tilde{f}$  ein akzeptierender Pfad für  $w$  in  $\tilde{A}$ .

Wir bauen einen akzeptierenden Pfad  $P = s \Rightarrow f$  für  $w$  in  $A$ .

Sei  $\tilde{e} = q \xrightarrow{a} q'$  eine Kante in  $\tilde{P}$ .

$\xrightarrow{\text{Def. } \tilde{\delta}} \exists p \in Q : p \in E(q) \wedge \delta(p, a) = q'$

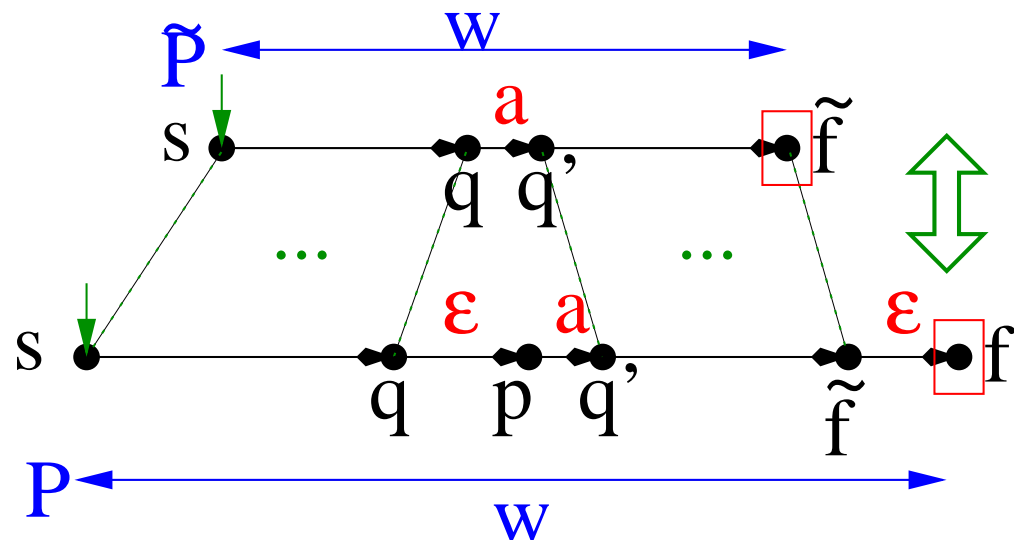
Ersetze  $\tilde{e}$  in  $P$  durch den Pfad

$$q \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} p \xrightarrow{a} q'.$$

Wähle ein  $f$  aus  $E(\tilde{f}) \cap F$ .

$$(E(\tilde{f}) \cap F \stackrel{\text{Def. } \tilde{F}}{\neq} \emptyset)$$

Hänge  $\tilde{f} \xrightarrow{\varepsilon} \dots \xrightarrow{\varepsilon} f$  an  $P$ .





**Satz:**  $L(\tilde{A}) = L(A)$

**Beweis für  $L(A) \subseteq L(\tilde{A})$**

Sei  $w \in L(A)$  beliebig.

Sei  $P = s \xrightarrow{w} f$  ein akzeptierender Pfad für  $w$  in  $A$ .

Wir bauen einen akzeptierenden Pfad  $\tilde{P} = s \xrightarrow{w} \tilde{f}$  für  $w$  in  $\tilde{A}$ .

Sei  $e = p \xrightarrow{a} q'$  ein beliebiger nicht- $\epsilon$ -Übergang in  $P$ .

Sei  $P_e = q \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} p \xrightarrow{a} q'$  der zu  $e$  gehörige **maximale** Teilpfad von  $P$ , der aus einer Folge von  $\epsilon$ -Übergängen besteht und mit  $e$  endet.

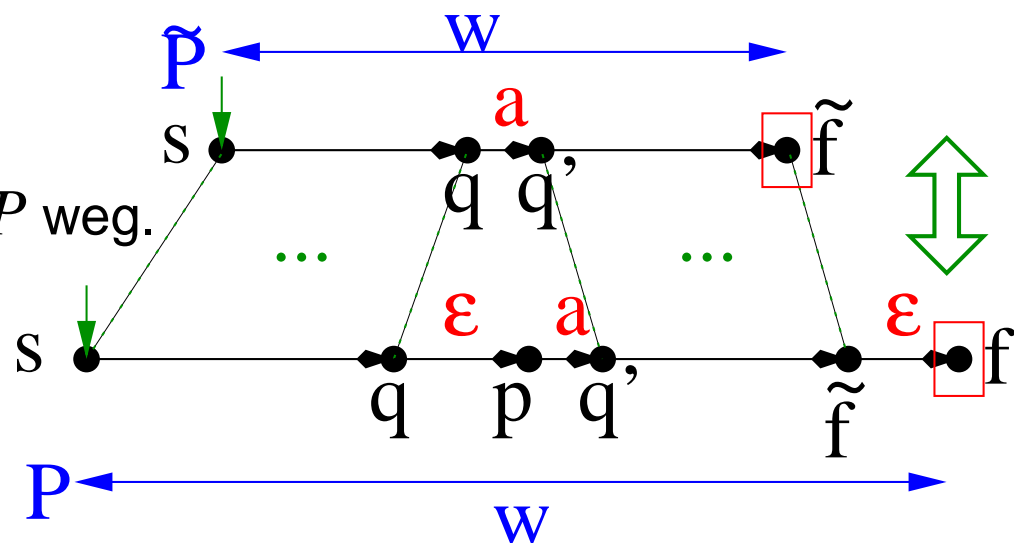
$\xrightarrow{\text{Def. } \tilde{\delta}} q' \in \tilde{\delta}(q, a)$ .

Ersetze  $P_e$  in  $P$  durch  $q \xrightarrow{a} q'$  in  $\tilde{P}$ .

Lasse  $\epsilon$ -Übergänge am Ende von  $P$  weg.

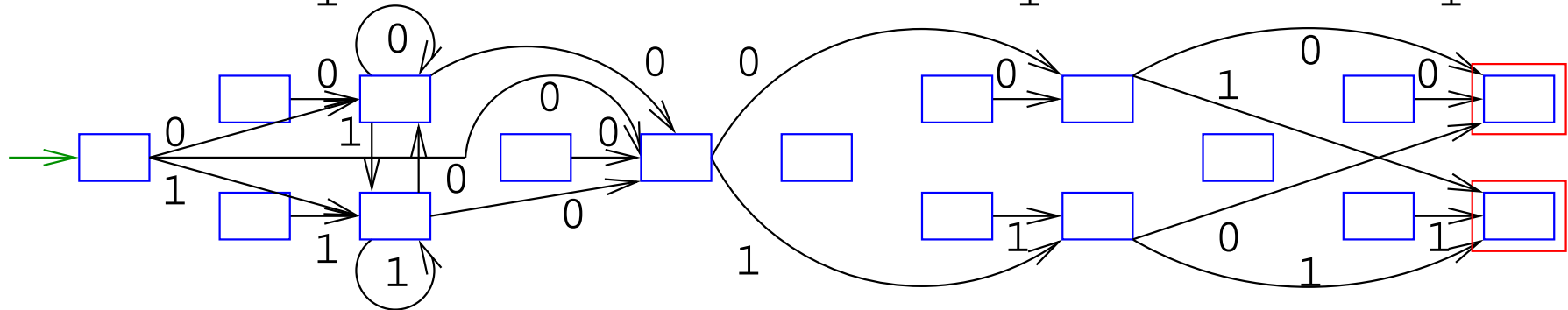
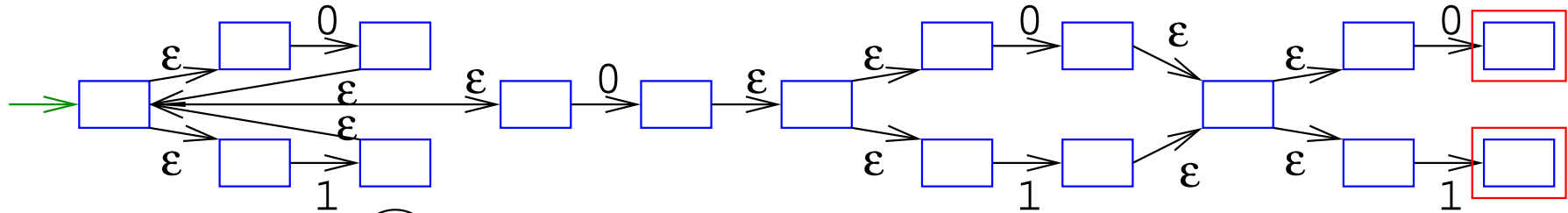
Betrachte  $\tilde{P} = s \Rightarrow \tilde{f}$

$\tilde{f} \in \tilde{F}$  (Def.  $\tilde{F}$ ).

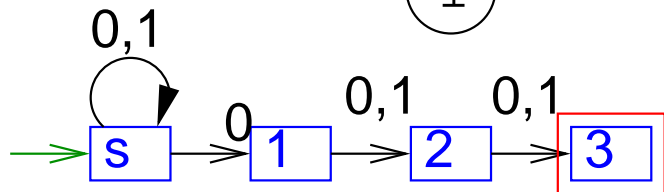
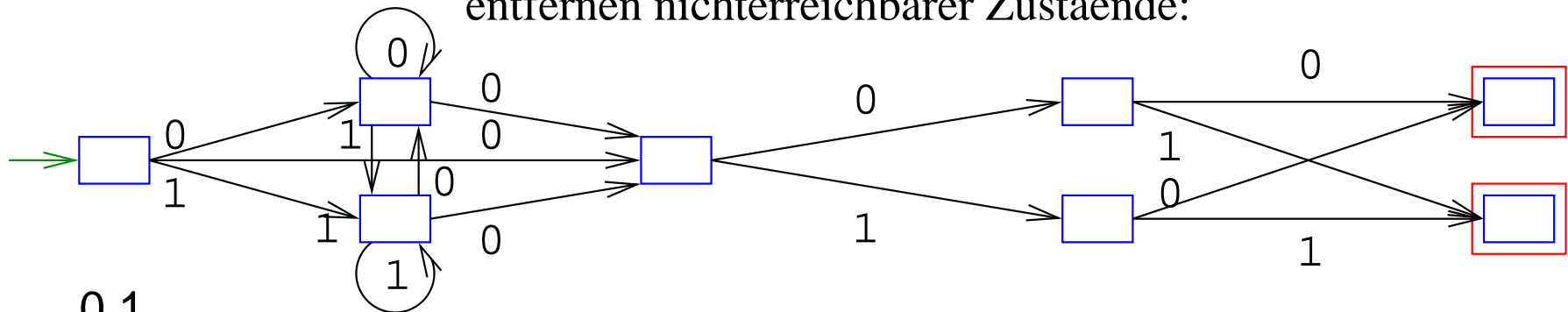




# Beispiel: $(0 \cup 1)^* 0(0 \cup 1)(0 \cup 1)$



entfernen nicht erreichbarer Zustände:



"handoptimiert"



## Erweiterung von $\delta$

Zustandsmengen:  $\delta : 2^Q \times \Sigma \rightarrow 2^Q$

$$\delta(M, a) := \bigcup_{p \in M} \delta(p, a)$$

Zustandsmengen und Eingabeworte:  $\delta : 2^Q \times \Sigma^* \rightarrow 2^Q$

$$\delta(M, \varepsilon) := M$$

— $\bar{\varepsilon}NEA!$

$$\delta(M, wa) := \delta(\delta(M, w), a) = \bigcup_{p \in \delta(M, w)} \delta(p, a)$$



**Lemma:**  $\delta(\{s\}, w) = \left\{ q \in Q : \exists \text{Pfad } P = s \xrightarrow{w} q \right\}$

**Beweis** durch Induktion über  $|w|$ :

$$\delta(\{s\}, \varepsilon) = \{s\}$$

$n \rightsquigarrow n + 1$ :

$$\delta(\{s\}, wa)$$

$$= \bigcup_{p \in \delta(\{s\}, w)} \delta(p, a) \quad (\text{Def. } \delta)$$

$$= \bigcup_{\left\{ p \in Q : \exists P = s \xrightarrow{w} p \right\}} \delta(p, a) \quad (\text{IV})$$

$$= \delta(\{s\}, wa) = \left\{ q \in Q : \exists \text{Pfad } P = s \xrightarrow{wa} q \right\} \quad \blacksquare$$



## $\bar{\text{NEA}} \rightarrow \text{DEA}$

Gegeben:  $\bar{\text{NEA}} A = (Q, \Sigma, \delta, s, F)$

**Lemma:**  $\delta(\{s\}, w) = \left\{ q \in Q : \exists \text{ Pfad } P = s \xrightarrow{w} q \right\}$

**Korollar:**  $L(A) = \{w \in \Sigma^* : \delta(\{s\}, w) \cap F \neq \emptyset\}$

**Korollar: (Potenzmengenkonstruktion)**

Der DEA  $A' := (2^Q, \Sigma, \delta, \{s\}, \{M \subseteq Q : M \cap F \neq \emptyset\})$  akzeptiert  $L(A)$ .

**Übung:** Entwerfen Sie einen Algorithmus, dessen Eingabe der NEA  $A$  und ein Wort  $w$  ist und der  $\delta(\{s\}, w)$  in Zeit  $O(|w| \cdot |\delta|)$  berechnet. Dabei ist  $|\delta|$  die Anzahl Einträge der Form  $p \in \delta(q, a)$ , die benötigt wird, um  $\delta$  zu definieren.



## Potenzmengenkonstruktion

$$A = (Q, \Sigma, \delta, s, F)$$

$$A' := (2^Q, \Sigma, \delta, \{s\}, F'), \quad F' := \{M \subseteq Q : M \cap F \neq \emptyset\}$$

Behauptung:  $L(A') = L(A)$

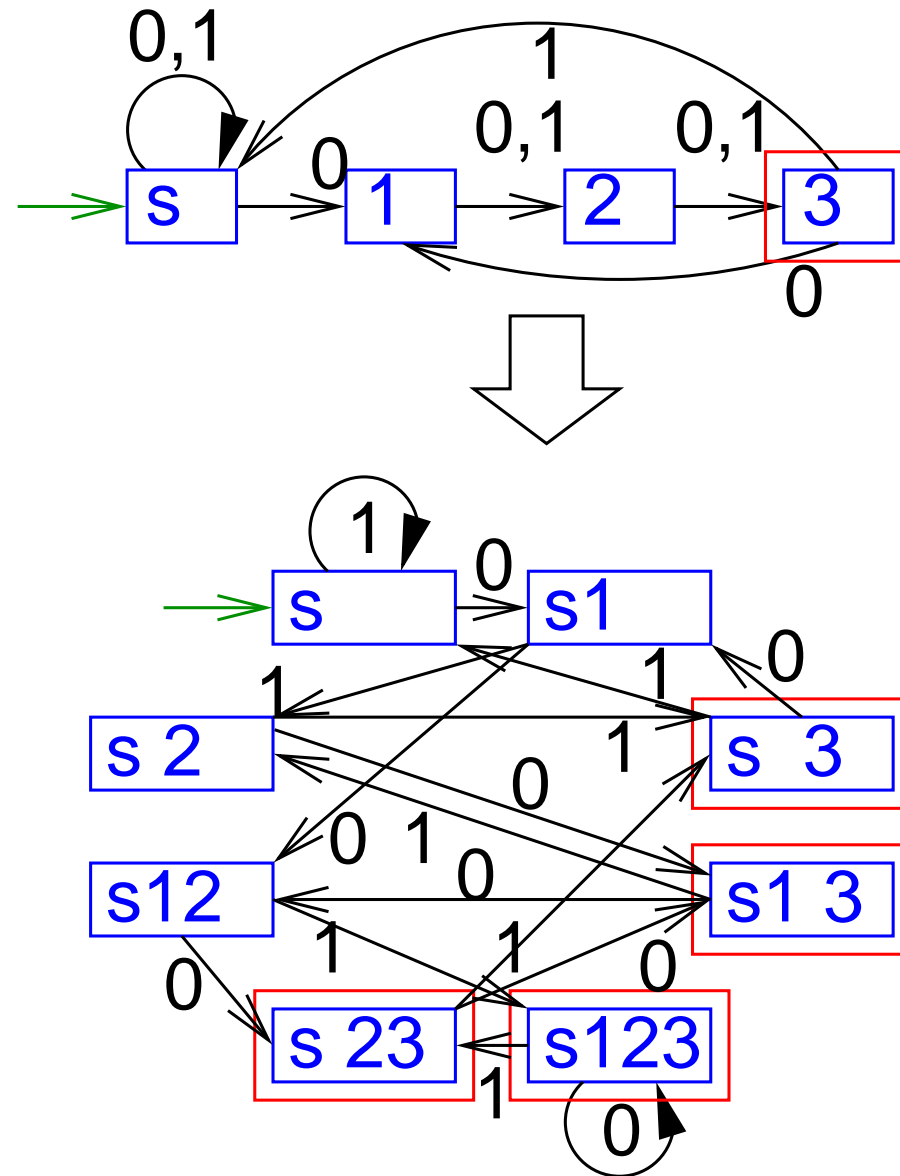
Beweis:

$$\begin{aligned} L(A) &= \left\{ w \in \Sigma^* : \exists f \in F : \exists P = s \xrightarrow{w} f \right\} \\ &= \left\{ w \in \Sigma^* : \left\{ q \in Q : \exists P = s \xrightarrow{w} q \right\} \cap F \neq \emptyset \right\} \\ &= \left\{ w \in \Sigma^* : \delta(\{s\}, w) \cap F \neq \emptyset \right\} \\ &= \left\{ w \in \Sigma^* : \delta(\{s\}, w) \in F' \right\} \\ &= L(A') \end{aligned}$$



# Beispiel

$q$	$\delta(q, 0)$	$\delta(q, 1)$
$s$	$s, 1$	$s$
$s, 1$	$s, 1, 2$	$s, 2$
$s, 2$	$s, 1, 3$	$s, 3$
$s, 3$	$s, 1$	$s$
$s, 1, 2$	$s, 1, 2, 3$	$s, 2, 3$
$s, 1, 3$	$s, 1, 2$	$s, 2$
$s, 2, 3$	$s, 1, 3$	$s, 3$
$s, 1, 2, 3$	$s, 1, 2, 3$	$s, 2, 3$





## Implementierungshinweise

Nur von  $\{s\}$  aus erreichbare Teilmengen erzeugen:

$Q' := \{\{s\}\}$  // states of  $A'$

Queue todo :=  $Q'$

**while**  $\exists M \in \text{todo}$  **do**

    todo := todo  $\setminus M$

**foreach**  $a \in \Sigma$  **do**

**if**  $M' = \delta(M, a) \notin Q'$  **then**

            insert  $M'$  into  $Q'$

            insert  $M'$  into todo

Laufzeit ist  $O(|Q'| \cdot |\delta|)$  bei Benutzung geeigneter Datenstrukturen.

Oft  $|Q'| \ll 2^{|Q|}$ !



# Anwendung Texte durchsuchen

Unix-Tool grep: `grep REGULAR-EXPRESSION FILE`

- Suche all Teilstrings in FILE, die in  $L(\text{REGULAR-EXPRESSION})$  sind
- Viel syntaktischer Zuckerguß: Bereiche a–g, vordefinierte Ausdrücke z.B. `:a1num:`, vorgegebene Anzahl Wiederholungen,...
- Aber alles ist leicht in einfache reg. Ausdrücke übersetzbar.
- Schnelle Ausführung durch Übersetzung in deterministische endliche Automaten.



## Anwendung Scanner-(Generatoren), `lex`, `flex`

**Eingabe:** Ein System von regulären Ausdrücken

**Ausgabe:** Ein endlicher Automat (C code),

**Laufzeit-Eingabe:** Das Programm als Zeichenkette

**Laufzeit-Ausgabe:** Das Programm als Folge von Token (Pakete) wie  
Zahl, Bezeichner, Schlüsselwort.

- zeitkritisch weil nur hier jedes einzelne Zeichen angefasst wird, Kommentare weggeworfen werden, Dezimalzahlen binär gewandelt werden,...
- normiert die Darstellung, durch Entfernen von Leerzeichen Varianten von Zahlrepräsentationen,...
- vereinfacht die nachfolgende Syntaxanalyse



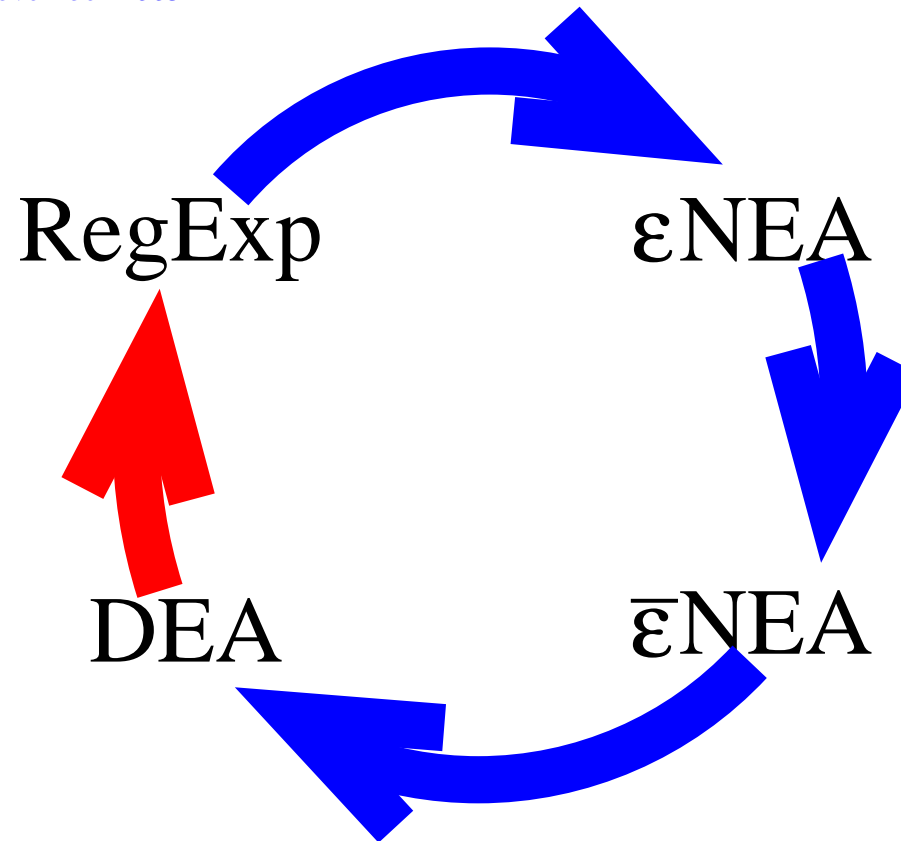
## Ähnliche Funktionalität

- Editoren, z.B. emacs
- Script-Sprachen wie Perl (berüchtigt?)
- `java.util.regex` Bibliothek
- C++ Boost.Regex Bibliothek
- .net framework
- Vorverarbeitung beim Parsing von xml Dokumenten



# DEA $\rightarrow$ RegExp

Ziel:



- Kein „Sprachenzoo“
- reguläre Ausdrücke sind universelles Interface
- reverse engineering (vielleicht)



**Beweis:** geg: DEA  $A = (\{1, \dots, n\}, \Sigma, \delta, s, F)$

ges: regulärer Ausdruck  $R$  mit  $L(A) = L(R)$ .

Für  $f \in F$  sei  $L_f = \{w \in \Sigma^* : \delta(s, w) = f\}$ .

Da  $L(A) = \bigcup_{f \in F} L_f$ , genügt es, einen RegExp für  $L_f$  zu finden.



geg: DEA  $A_f = (\{1, \dots, n\}, \Sigma, \delta, s, \{f\})$

ges: regulärer Ausdruck  $R$  mit  $L_f = L(A_f) = L(R)$ .

Sei  $L_{ij} := L((\{1, \dots, n\}, \Sigma, \delta, i, \{j\}))$

Also insbesondere  $L_{sf} = L_f$ .

$L_{ij}^m := \left\{ w \in \Sigma^* : \exists \text{Abarbeitungspfad } i \xrightarrow{w} j = iPj \text{ mit } P \in \{1, \dots, m\}^* \right\}$

Beachte, dass  $L_{ij} = L_{ij}^n$ .

Wir konstruieren  $L_{ij}^m$  induktiv

aus den reguläreren Ausdrücken für kleinere  $m$ .



$$L_{ij}^m := \left\{ w \in \Sigma^* : \exists \text{Abarbeitungspfad } i \xrightarrow{w} j = iPj \text{ mit } P \in \{1, \dots, m\}^* \right\}$$

geg: reguläre Ausdrücke  $R_{ij}^k, k < m$  mit  $L(R_{ij}^k) = L_{ij}^k$

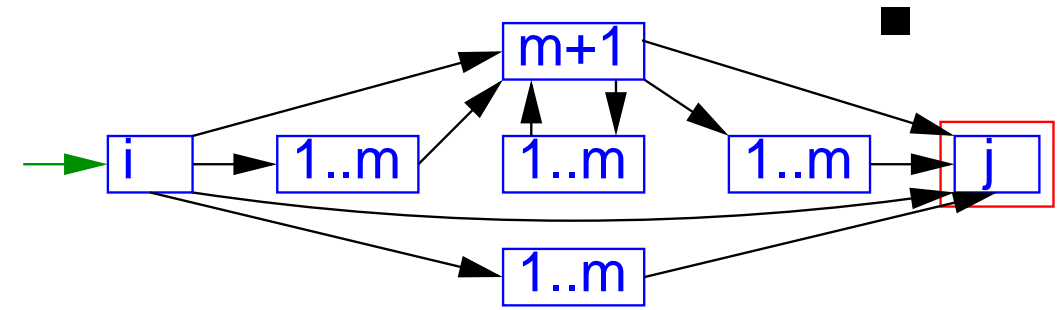
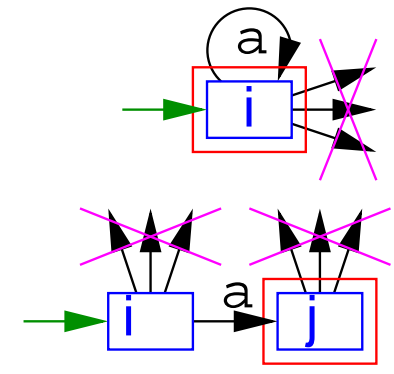
ges:  $R_{ij}^m$  mit  $L(R_{ij}^m) = L_{ij}^m$

**Fall**  $m = 0, i = j$ :  $L_{ii}^0 = \{a \in \Sigma : \delta(i, a) = i\} \cup \epsilon$

**Fall**  $m = 0, i \neq j$ :  $L_{ij}^0 = \{a \in \Sigma : \delta(i, a) = j\}$

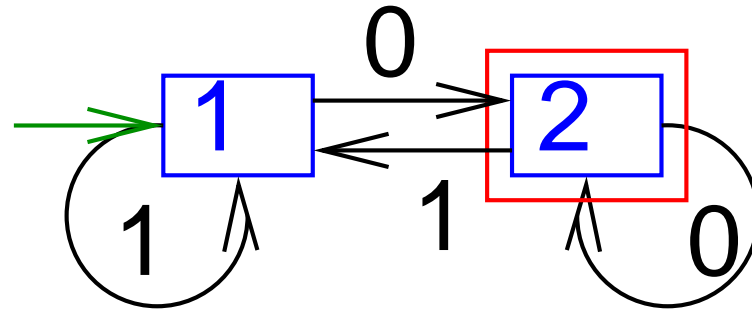
**Fall**  $m \rightsquigarrow m + 1$ :

$$R_{ij}^{m+1} = R_{ij}^m \cup R_{i,m+1}^m \cdot (R_{m+1,m+1}^m)^* \cdot R_{m+1,j}^m$$





# Beispiel



$$R_{11}^0 = 1 \cup \epsilon$$

$$R_{22}^0 = 0 \cup \epsilon$$

$$R_{12}^0 = 0$$

$$R_{21}^0 = 1$$

$$\begin{aligned} R_{12}^1 &= R_{12}^0 \cup R_{11}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 \\ &= 0 \cup (1 \cup \epsilon) \cdot (1 \cup \epsilon)^* \cdot 0 \\ &= 1^*0 \end{aligned}$$

$$\begin{aligned} R_{22}^1 &= R_{22}^0 \cup R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{12}^0 \\ &= 0 \cup \epsilon \cup 1 \cdot (1 \cup \epsilon)^* \cdot 0 \\ &= 1^*0 \cup \epsilon \end{aligned}$$

$$\begin{aligned} R_{12}^2 &= R_{12}^1 \cup R_{12}^1 \cdot (R_{22}^1)^* \cdot R_{12}^1 \\ &= 1^*0 \cup 1^*0 \cdot (1^*0 \cup \epsilon)^* \cdot (1^*0 \cup \epsilon) \\ &= 1^*0(1^*0)^* \end{aligned}$$

$$L(R_{12}^2) = L_{12}^2 = L_{12} = L_2, \text{ wobei } F = \{2\}.$$



## **2.3 Zustandsminimierung etc.**

Idee: arbeite direkt mit  $L$  ohne Bezug zu einem konkreten Automaten.

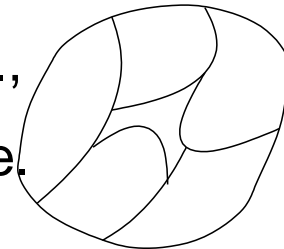


## Zur Erinnerung: Äquivalenzrelationen

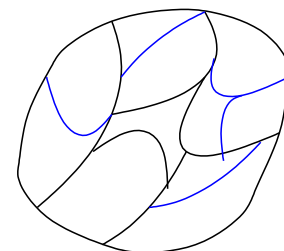
Eine Relation  $R \subseteq Y \times Y$  heisst Äquivalenzrelationen gdw. sie:

- reflexiv  $\forall x : xRx$
- transitiv  $\forall xyz : xRy \wedge yRz \rightarrow xRz$
- symmetrisch ist.  $\forall xy : xRy \rightarrow yRx$

**Äquivalenzklasse:**  $[x] = \{y : xRy\}$ . Äq.-Klassen sind gleich oder disjunkt und induzieren eine Partitionierung von  $Y$ , d.h., jedes Element von  $Y$  gehört zu genau einer Äq.-Klasse.



**Index:**  $\text{index}(R) := |\text{Äquivalenzklassen}| = |\{[x] : x \in Y\}|$





**Verfeinerung:**  $R$  verfeinert  $R'$  gdw  $R \subseteq R'$

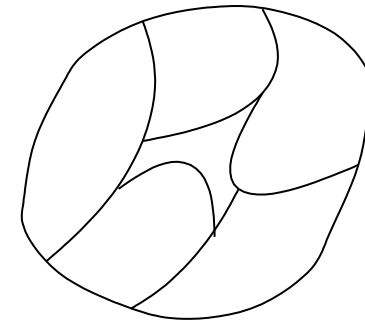
**Lemma:**  $R$  verfeinert  $R' \rightarrow \forall$  Äquivalenzklassen  $[x]_R : [x]_R \subseteq [x]_{R'}$

**Beweis:**

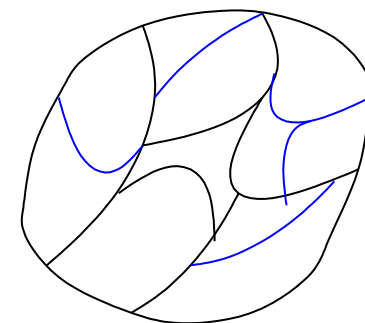
$$y \in [x]_R \leftrightarrow (y, x) \in R$$

$$\xrightarrow{R \subseteq R'} (y, x) \in R'$$

$$\leftrightarrow y \in [x]_{R'}$$



**Korollar:**  $R$  verfeinert  $R' \rightarrow \text{index}(R) \geq \text{index}(R')$





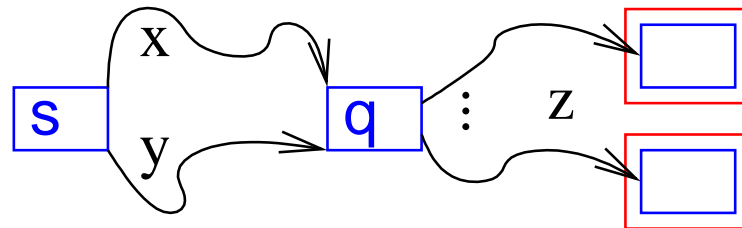
# Nerode Relation

Zur Sprache  $L$  ist die **Nerode Relation**

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

Idee: Äquivalenzklassen entsprechen Zuständen.

Wieso? Genauer!





# DEAs induzieren Äquivalenzrelationen

Sei  $M = (Q, \Sigma, \delta, s, F)$  DEA mit  $L(M) = L$ .

$$R_M := \{(x, y) \in \Sigma^* \times \Sigma^* : \delta(s, x) = \delta(s, y)\}.$$

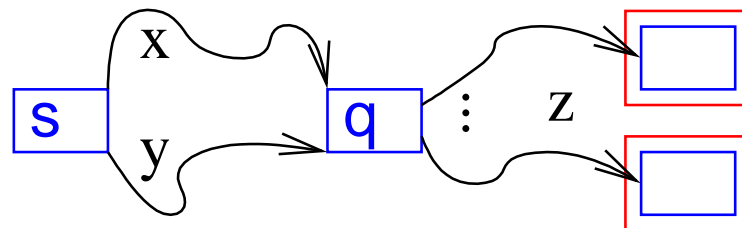
Äquivalenzrelation! Eine Äquivalenzklasse pro (von  $s$  erreichbarem) Zustand.

**Lemma 1:**  $R_M$  **verfeinert** die Neroderelation  $R_L =$

$$\{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

**Beweis:** z.Z.

$$\forall (x, y) \in \Sigma^* \times \Sigma^* : \delta(s, x) = \delta(s, y) \rightarrow \forall z : xz \in L \Leftrightarrow yz \in L$$





## Unendlicher Index der Neroderelation

**Beobachtung:**  $\text{index}(R_L) = \infty \rightarrow L$  ist nicht regulär.

**Beweis:** Annahme  $L$  ist regulär.

$\rightarrow \exists$  DEA  $M = (Q, \Sigma, \delta, s, F) : L(M) = L$ .

$\rightarrow R_M$  verfeinert  $R_L$ .

$\rightarrow |Q| \geq \text{index}(R_M) \geq \text{index}(R_L) = \infty$ .

Widerspruch

**Ab jetzt:**  $\text{index}(R_L) < \infty$ .



# Äquivalenzklassenautomat

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

Idee: wenn die Äquivalenzklassen  $[w_1], \dots, [w_k]$  von  $R_L$  zu den Zuständen eines DEA  $M_{\equiv}$  korrespondieren, so ist dieser wegen Lemma 1 DER **zustandsminimale** Automat für  $L$ .

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv}) \text{ mit}$$

$$F_{\equiv} := \{[w] : w \in L\} \text{ und}$$

$$\delta_{\equiv}([w], a) := [wa].$$

**Lemma:**  $\delta_{\equiv}$  ist wohldefiniert

$$\text{Lemma: } \delta_{\equiv}([\varepsilon], w) = [w]$$

$$\text{Lemma: } L(M_{\equiv}) = L$$



# Äquivalenzklassenautomat

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\epsilon], F_{\equiv}) \text{ mit}$$

$$F_{\equiv} := \{[w] : w \in L\} \text{ und}$$

$$\delta_{\equiv}([w], a) := [wa].$$

**Lemma:**  $\delta_{\equiv}$  ist wohldefiniert

$$\text{Also } xR_Ly \rightarrow \forall a \in \Sigma : xaR_Lya$$

$$xR_Ly \rightarrow \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L$$

$$\rightarrow \forall az \in \Sigma^* : x(az) \in L \Leftrightarrow y(az) \in L$$

$$\Leftrightarrow \forall a \in \Sigma : \forall z \in \Sigma^* : (xa)z \in L \Leftrightarrow (ya)z \in L$$

$$\rightarrow \forall a \in \Sigma : xaR_Lya$$

*Rechtsinvarianz*

insbesondere

□



# Äquivalenzklassenautomat

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv}) \text{ mit}$$

$$F_{\equiv} := \{[w] : w \in L\} \text{ und}$$

$$\delta_{\equiv}([w], a) := [wa].$$

**Lemma:**  $\delta_{\equiv}([x], y) = [xy]$

Induktion über  $|y|$ :

$$\delta_{\equiv}([x], \varepsilon) = [x].$$

$$\delta_{\equiv}([x], aw) \stackrel{\text{Def. } \delta_{\equiv}}{=} \delta_{\equiv}(\delta_{\equiv}([x], a), w) \stackrel{\text{Def. } \delta_{\equiv}}{=} \delta_{\equiv}([xa], w) \stackrel{IV}{=} [xaw]. \quad \square$$



## Äquivalenzklassenautomat: akzeptiert $L$

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}$$

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv}) \text{ mit}$$

$$F_{\equiv} := \{[w] : w \in L\} \text{ und}$$

$$\delta_{\equiv}([w], a) := [wa].$$

**Lemma:**  $L(M_{\equiv}) = L$ .

$$w \in L(M_{\equiv})$$

$$\Leftrightarrow \delta_{\equiv}([\varepsilon], w) \in \{[w] : w \in L\}$$

$$\Leftrightarrow [w] \in \{[w] : w \in L\}$$

$$\Leftrightarrow w \in L.$$

Def.  $M_{\equiv}$

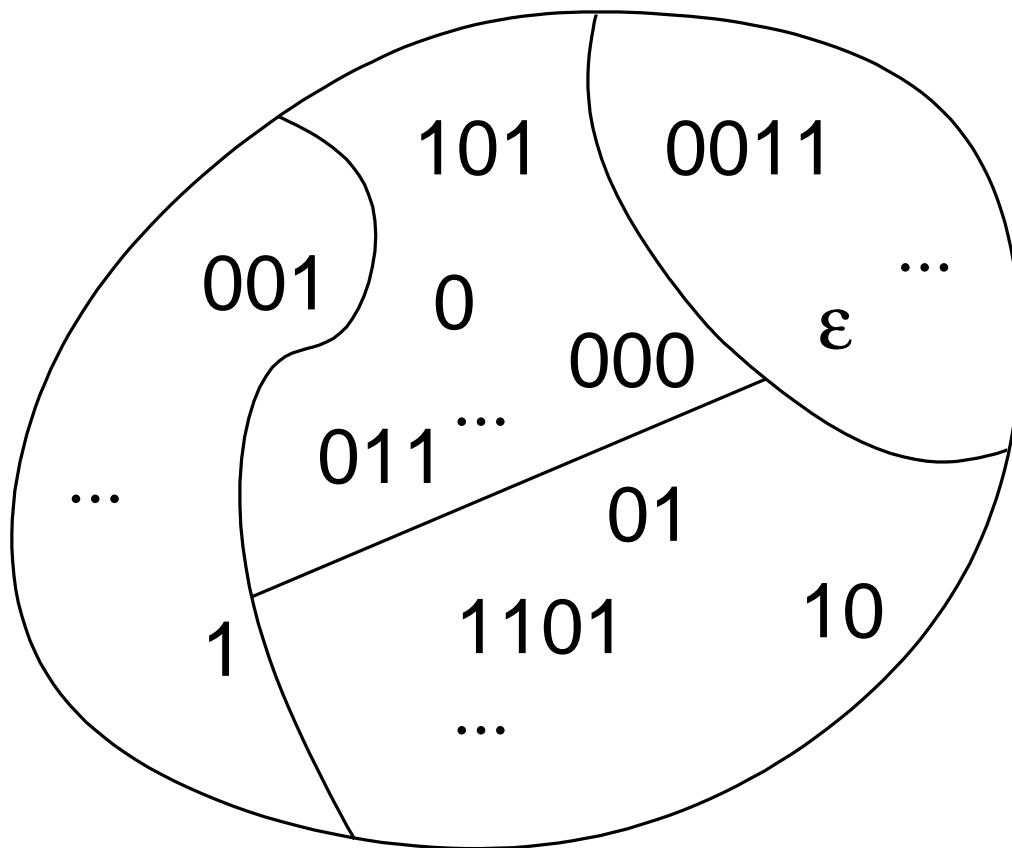
voriges Lemma

Äq.Klassen sind ganz oder garnicht in  $L$



# Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen

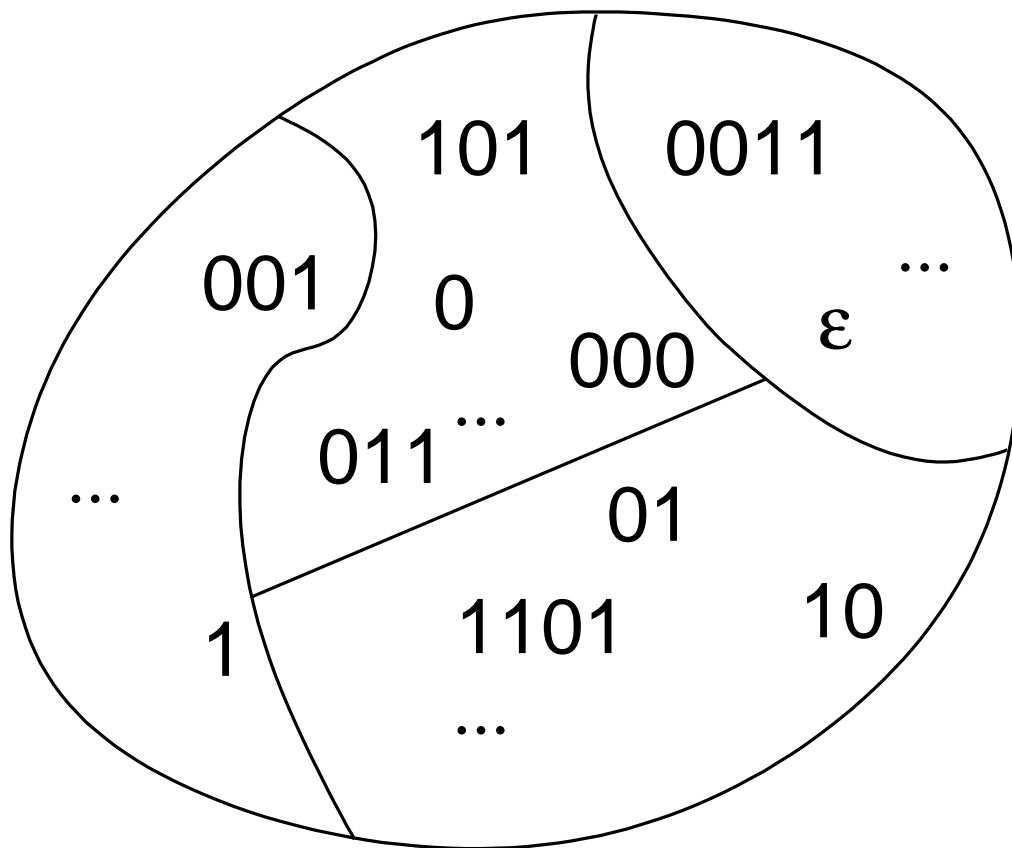


**Äquivalenzklassen?**



# Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen



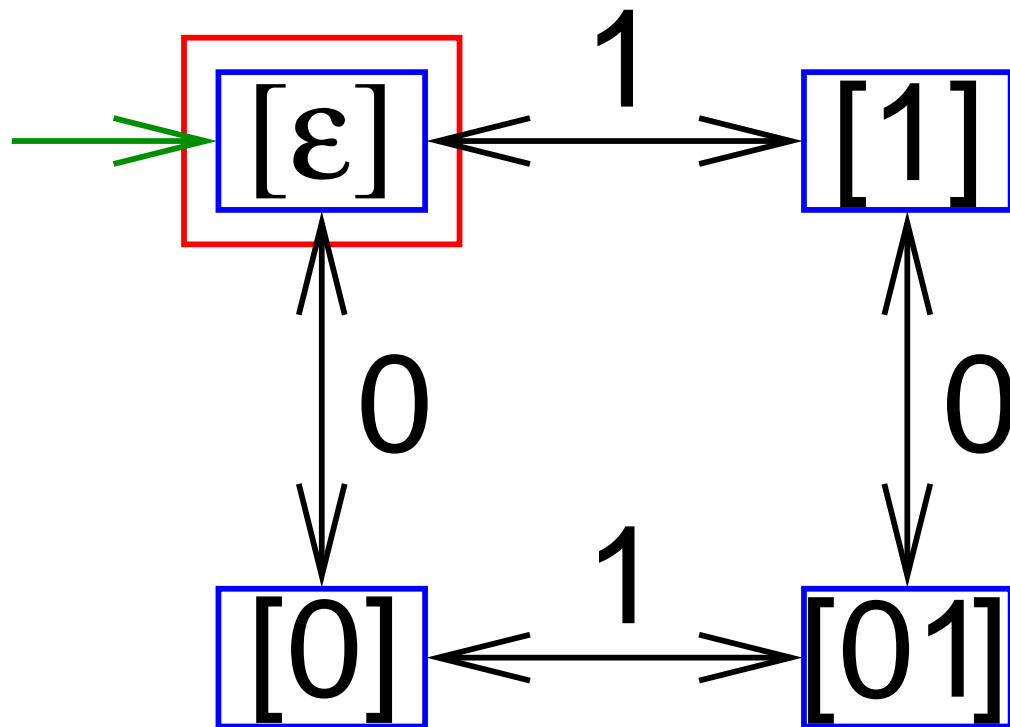
**Äquivalenzklassen:**

$[\epsilon], [0], [1], [01]$



# Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen





## Satz von Nerode

Sei

$$R_L := \{(x, y) \in \Sigma^* \times \Sigma^* : \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\}.$$

$L$  nicht regulär  $\rightarrow \text{index}(R_L) = \infty$

$L$  regulär  $\rightarrow$  Sei

$$M_{\equiv} := (\{[w_1], \dots, [w_k]\}, \Sigma, \delta_{\equiv}, [\varepsilon], F_{\equiv}), \text{ und}$$

$$M = (Q, \Sigma, \delta, s, F) \text{ beliebiger DEA mit } L(M) = L.$$

Es gilt  $L(M_{\equiv}) = L$  und  $R_M$  ist Verfeinerung von  $R_L$ .

### Korollar:

Alle zustandsminimalen Automaten für  $L$  sind isomorph zu  $M_{\equiv}$ .

(gleich bis auf Zustandsumbenennung)



# Konstruktion eines zustandsminimalen Automaten

## Vorverarbeitung

Entferne Zustände, die von  $s$  aus **nicht erreichbar** sind.

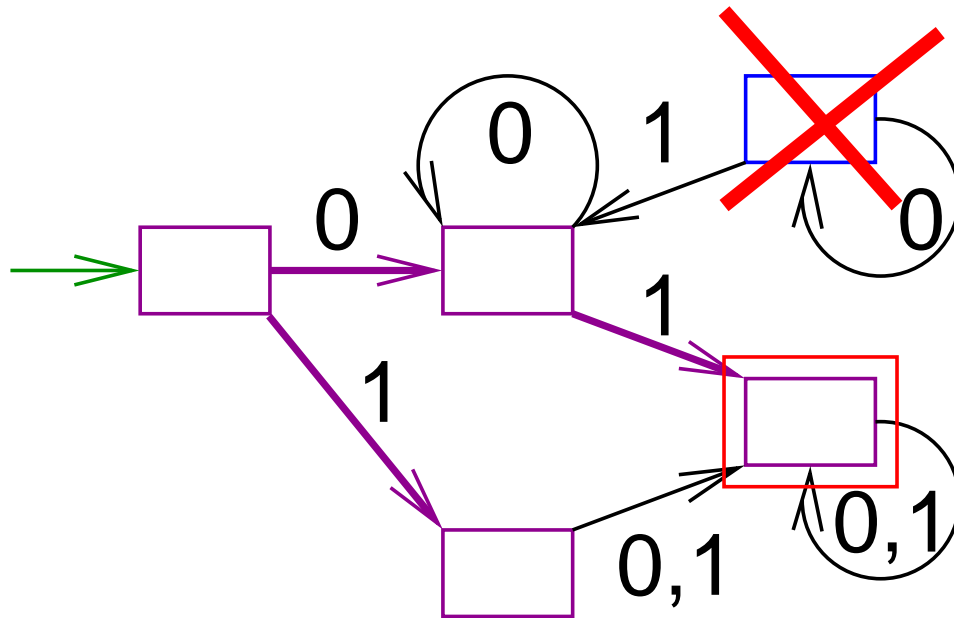
Algorithmus: **Tiefensuche** im Graph  $G_A$  von  $s$  aus.

Markiere alle erreichten Zustände.

Entferne nicht erreichte Zustände.



# Vorverarbeitung — Beispiel



Kante im  
Tiefensuchbaum

 erreicht  
erreichter Zustand



# Äquivalente Zustände

Idee: betrachte DEA  $M = (Q, \Sigma, \delta, s, F)$  (ohne nichterr. Zustände)

$M$  nicht zustandsminimal  $\rightarrow$

$R_M$  verfeinert  $R_L \rightarrow$

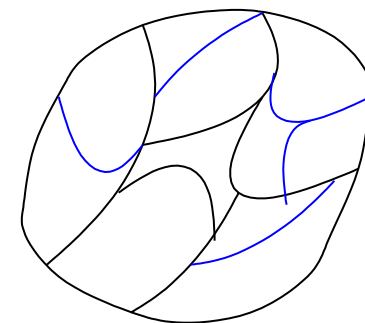
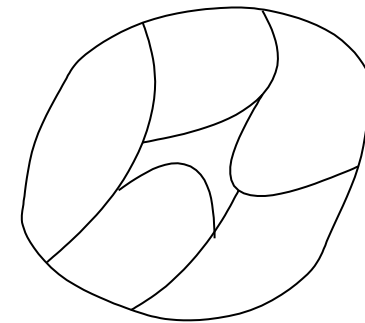
$\exists q \neq r \in Q : \text{Äq-Klasse}(q) \cup \text{Äq-Klasse}(r) \subseteq M$

für eine Äq-Klasse von  $M$  von  $R_L$

solche  $q, r$  nennen wir **äquivalent**,  $q \equiv r$ .

Insbesondere:

$\forall w \in \Sigma^* : \delta(q, w) \in F \Leftrightarrow \delta(r, w) \in F$





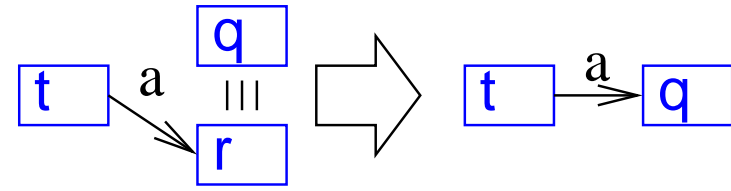
# Entfernen äquivalenter Zustände

Betrachte  $q \neq r \in Q : q \equiv r$  mit  $r \neq s$

Entferne  $r$ :

$M' := (Q \setminus \{r\}, \Sigma, \delta', s, F \setminus \{r\})$  mit

$$\delta'(t, a) := \begin{cases} q & \text{falls } \delta(t, a) = r \\ \delta(t, a) & \text{sonst} \end{cases}$$



**Lemma:**  $L(M') = L$

Beweis: Übung



# Zustandsminimierung

Erster Versuch:

**Function** minDEA( $M$ )

remove states not reachable from  $s$

**while**  $\exists q, r \in Q : q \equiv r \wedge q \neq r \wedge q \neq s$  **do**

remove  $q$  from  $M$

**return**  $M$

**Problem:** wie findet man äquivalente Zustände?

$q \equiv r$  gdw.  $\forall z \in \Sigma^* : \delta(q, z) \in F \Leftrightarrow \delta(r, z) \in F$

Allquantifizierung über **unendliche** Menge!

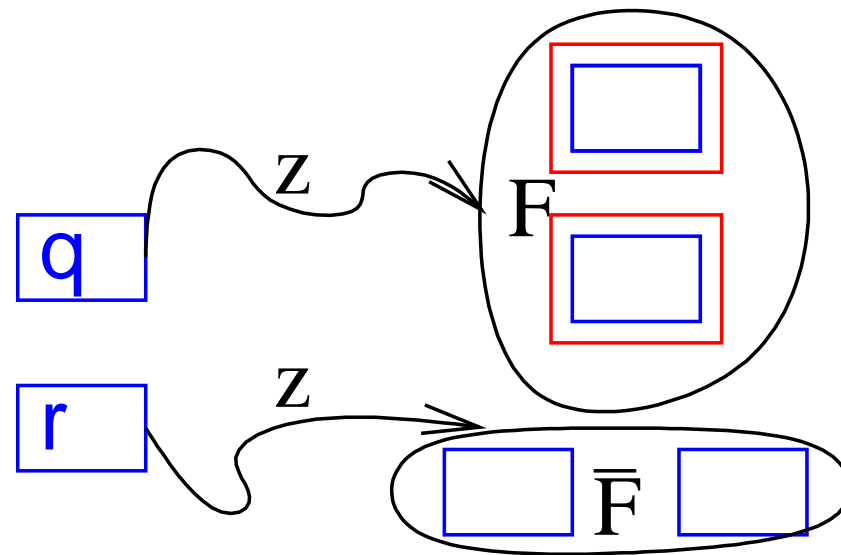


## Nichtäquivalenz von Zuständen

$q \equiv r$  gdw.  $\forall z \in \Sigma^* : \delta(q, z) \in F \Leftrightarrow \delta(r, z) \in F$

$q \not\equiv r$  gdw.  $\exists z \in \Sigma^* : \delta(q, z) \in F \not\Leftrightarrow \delta(r, z) \in F$

$z$  **bezeugt** Nichtäquivalenz.



Problem: finde Zeugen für Nichtäquivalenz



## Kürzeste Zeugen für Nichtäquivalenz

$\varepsilon$  bezeugt  $q \not\equiv r$  falls  $q \in F, r \notin F$  oder  $q \notin F, r \in F$ .

Sei  $w = aw'$  kürzester Zeuge für  $q \not\equiv r$ .

**Beobachtung:**  $w'$  ist Zeuge für  $q' := \delta(q, a) \not\equiv \delta(r, a) =: r'$

**Lemma:**  $w'$  ist **kürzester** Zeuge für  $q' \not\equiv r'$

Widerspruchsbeweis: Annahme:

$w''$  ist kürzerer Zeuge für  $q' \not\equiv r'$

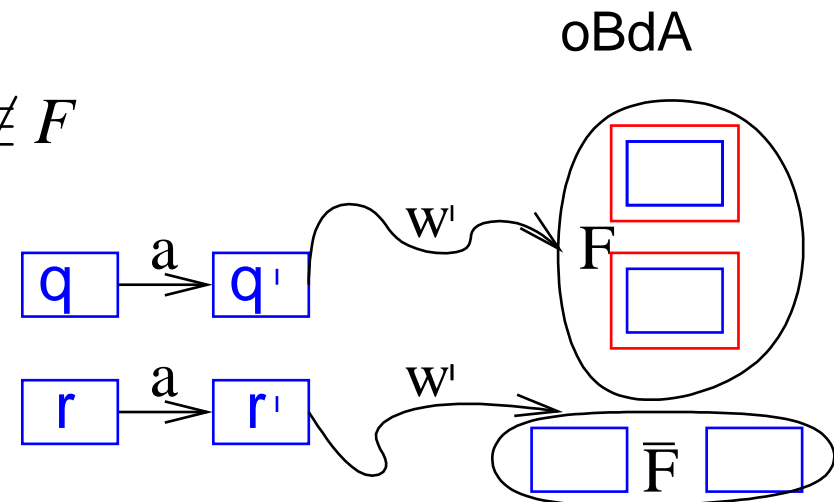
$\rightarrow \delta(q', w'') \in F \wedge \delta(r', w'') \notin F$

$\rightarrow \delta(\delta(q, a), w'') \in F \wedge \delta(\delta(r, a), w'') \notin F$

$\rightarrow \delta(q, aw'') \in F \wedge \delta(r, aw'') \notin F$

$\rightarrow aw''$  ist kürzerer Zeuge für  $q \not\equiv r$

Widerspruch





## Kürzeste Zeugen für Nichtäquivalenz

$\varepsilon$  bezeugt  $q \not\equiv r$  falls  $q \in F, r \notin F$ .

Sei  $w = aw'$  kürzester Zeuge für  $q \not\equiv r$ .

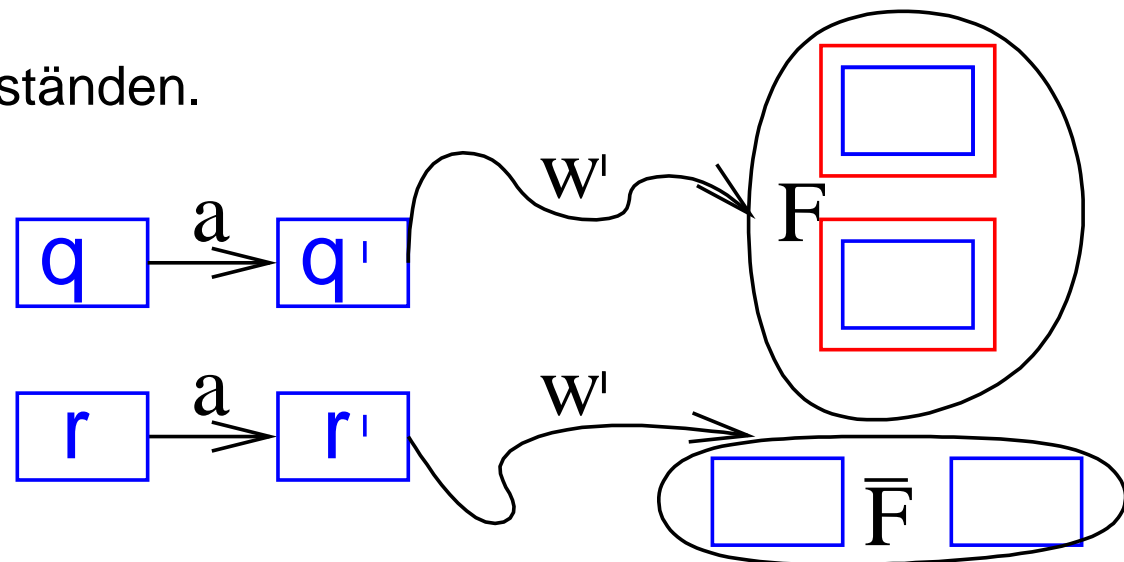
**Lemma:**  $w'$  ist kürzester Zeuge für  $q' := \delta(q, a) \not\equiv \delta(r, a) =: r'$

**Folgerung:**

**kürzeste** Zeugen lassen sich systematisch und **effizient** erzeugen.

$\rightsquigarrow$  alle Nichtäquivalenzen

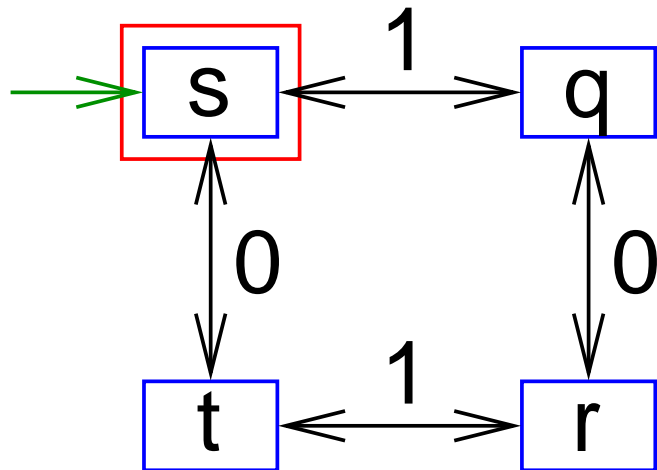
$\rightsquigarrow$  Äquivalenzklassen von Zuständen.





# Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit gerader Zahl an Einsen und gerader Zahl an Nullen



$0 = 0\varepsilon$  ist kürzester Zeuge für  $t \neq r$ .

$\rightsquigarrow \varepsilon$  ist kürzester Zeuge für  $s = \delta(t, 0) \neq \delta(r, 0) = q$ .



Sei  $w = aw'$  kürzester Zeuge für  $q \neq r$ .

**Lemma:**  $w'$  ist kürzester Zeuge für  $q' := \delta(q, a) \neq \delta(r, a) =: r'$

Sei  $M$  die Menge aller nichtäquivalenten Zustandspaare für die es Zeugen der Länge  $\leq k$  gibt.

Für welche Zustandspaare  $\{q, r\}$  gibt es Zeugen der Länge  $k + 1$ ?

Lemma  $\rightsquigarrow \exists a \in \Sigma : \{\delta(q, a), \delta(r, a)\} \in M$



## Ein Einfacher Algorithmus

$M := \emptyset$  // marked pairs

$M' := \{\{q, r\} \subseteq Q : q \in F \not\Rightarrow r \in F\}$  // pairs to be marked next

**while**  $M' \neq \emptyset$  **do**

$M := M \cup M'$

$M' := \{\{q, r\} \subseteq Q : \exists a \in \Sigma : \{\delta(q, a), \delta(r, a)\} \in M\} \setminus M$

**Gesamtzeit:**  $O(|\Sigma| \cdot |Q|^3)$

Initialisierung:  $O(|Q|^2)$

Ein Schleifendurchlauf:  $O(|\Sigma| \cdot |Q|^2)$

Wieviel **Schleifendurchläufe**? Sicherlich  $\leq |Q|^2$ .

Genauere Betrachtung:  $\leq |Q|$  **Schleifendurchläufe**



## Zustandsminimierung in Zeit $O(|\Sigma| \cdot |Q| \log |Q|)$

[Hopcroft 1971]. Geschickte Datenstrukturen.

Leichte Vereinfachung:

[Blum, Minimization of finite automata in  $O(n \log n)$  time, Inf. Proc. Letters, 1996.]



## Wieso Zustandsminimierung

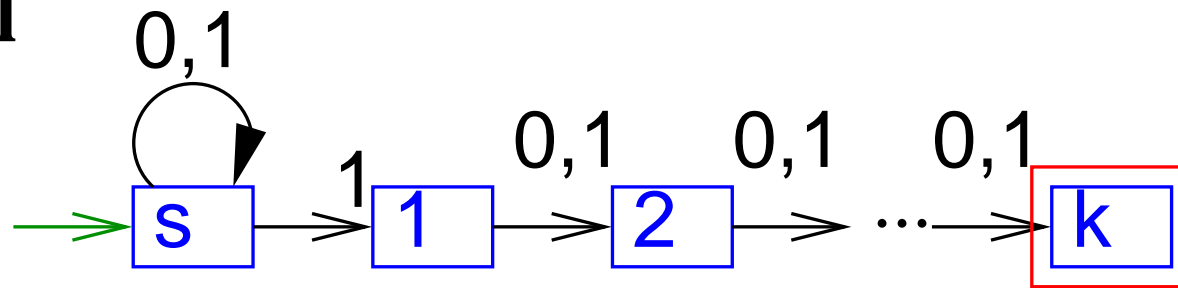
- Minimiert Platz für **Zustandsübergangstabelle**
- Minimalautomat **eindeutig**  $\rightsquigarrow$  wir lernen etwas über die akzeptierte Sprache.

**Aber**, wenn  $\delta$  als **Schaltkreis** oder **Programm** repräsentiert ist, wollen wir deren Größe und Zeitaufwand optimieren.

Im allgemeinen schwierig  $\rightsquigarrow$  aktives Forschungsgebiet



# Beispiel

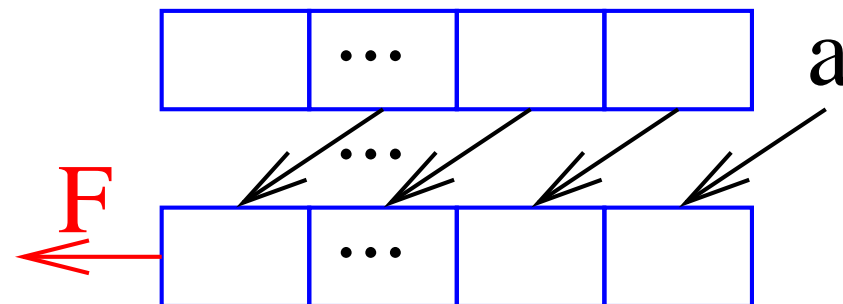


$$L = \{0, 1\}^* 1 \{0, 1\}^{k-1}$$

der Minimalautomat hat  $2^k$  Zustände.

$$(\{0, \dots, 2^k - 1\}, \{0, 1\}, \delta, 0, F)$$

$$\delta(q, a) = 2q + a, q \in F \Leftrightarrow q[k-1] = 1$$





# Nachweis von Nichtregularität

Gibt es Sprachen, die nicht regulär sind?



# Nerode Relation

Hinreichende und notwendige Bedingung:  $\text{index}(R_L) = \infty$

**Beispiel:**  $L = \{a^n b^n\}$

Behauptung:  $\forall k \geq 1, j \neq k \geq 1 : [a^k b] \neq [a^j b]$

$$[a^k b] = \{a^k b, a^{k+1} b b, \dots\} = \{a^{k+i} b^{i+1}\}$$

also immer  $k - 1$  mehr a's als b's.

Folglich sind  $[a^k b]$  und  $[a^j b]$  disjunkt

□

Problem: unhandliches Kriterium



# Pumping Lemma für Reguläre Sprachen

$L$  regulär

$$\rightarrow \exists n \in \mathbb{N} : \forall w \in L : |w| > n$$

$$\rightarrow \exists u, v, x : w = uvx \wedge |v| \geq 1 \wedge |uv| < n \wedge$$

$$\forall i \in \mathbb{N}_0 : uv^i x \in L$$

In Worten:

Hinreichend lange Worte einer regulären Sprache lassen sich durch

**Wiederholung** eines nichttrivialen **Mittelteiles** „aufpumpen“.



# Beweis Pumping Lemma

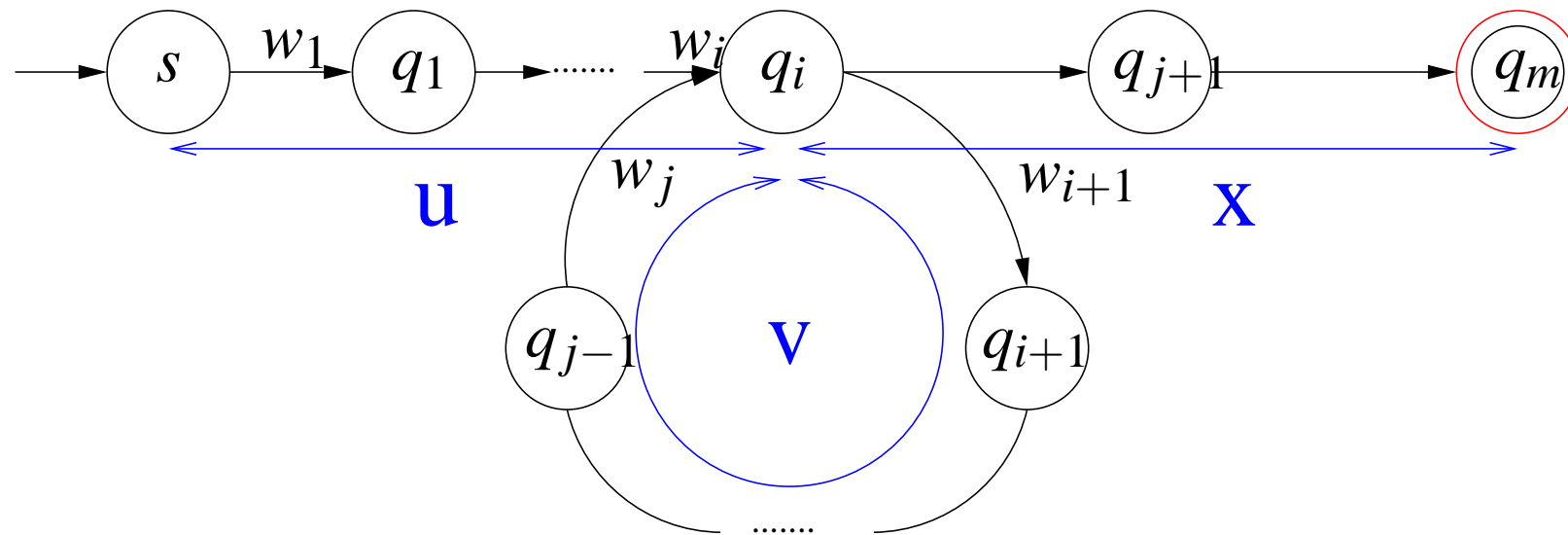
$L$  regulär  $\rightarrow \exists n \in \mathbb{N} : \forall w \in \Sigma^* : |w| > n \rightarrow \exists u, v, x :$

$$w = uvx \wedge |v| \geq 1 \wedge |uv| < n \wedge \forall i \in \mathbb{N}_0 : uv^i x \in L$$

**Beweis:** Sei  $A = (Q, \Sigma, \delta, s, F)$  DEA mit  $L(A) = L$ .

Sei  $n = |Q|$  und  $w \in L$  mit  $|w| = m > n$  beliebig.

Seien  $q_1, \dots, q_m$  die durchlaufenen Zustände.



$uv^i x$  wird ebenfalls akzeptiert





**Beispiel:**  $L = \{a^k b^k : k \in \mathbb{N}\}$

Annahme  $L$  regulär.

Sei  $n$  der Wert aus dem Pumping Lemma und betrachte

$w = a^n b^n = uvx$  so dass laut Pumping Lemma  $ux \in L$ .

Fall  $v = a^k$ :  $uvvx = a^{n+k} b^n \notin L$ . Widerspruch.

Fall  $v = b^k$ :  $uvvx = a^n b^{n+k} \notin L$ . Widerspruch.

Fall  $v = a^i b^j$ :  $uvvx = a^{n+i} b^j a^i b^{j+n} \notin L$ . Widerspruch.



## Beispiel: Wohlgeformte **Klammerausdrücke** $L_{()}$

Annahme  $L$  regulär.

Sei  $n$  der Wert aus dem Pumping Lemma und betrachte

$w = ({}^n)^n = uvx$  so dass laut Pumping Lemma

$ux \in L_{()}$  sowie  $|v| > 1$  und  $|uv| < n$ .

Dann ist  $v = ({}^i$

und  $ux = ({}^{n-i})^n \notin L_{()}$  Widerspruch.



**Beispiel:**  $L = \{1^k : k > 0\} \cup \{0^j 1^{k^2} : j \geq 0, k \geq 0\}$

Für  $n = 1$  ergibt sich kein Widerspruch zum Pumping Lemma!

Trotzdem ist  $L$  **nicht** regulär!

Fazit: Das Pumping Lemma ist nicht in jedem Fall geeignet

Nichtregularität zu zeigen.



## Abgeschlossenheitseigenschaften

Seien  $L, L'$  reguläre Sprachen.

Dann sind folgende Sprachen ebenfalls regulär:

$L \cup L', L^*, L \cdot L'$ : nach Definition.

$\bar{L} := \Sigma^* \setminus L$ : Betrachte DEA  $A = (Q, \Sigma, \delta, s, F)$  mit  $L(A) = L$ .

Sei  $\bar{A} := (Q, \Sigma, \delta, s, Q \setminus F)$ . Dann gilt  $L(\bar{A}) = \bar{L}$ .

$$L \cap L' = \overline{\bar{L} \cup \bar{L}'} \quad (\text{De Morgan})$$

$$L \setminus L' = L \cap \bar{L}'$$

$L^R$ : (Spiegelung) Übung. Hinweis: Induktion über reguläre Ausdrücke.



# Produktautomat

## Konstruktion eines DEA für Mengenoperationen

$L$  und  $L'$  seien reguläre Sprachen definiert durch DEAs

$$A = (Q, \Sigma, \delta, s, F),$$

$$A' = (Q', \Sigma, \delta', s', F').$$

Idee: Ein Automat  $A_{\times}$  emuliert das Verhalten von  $A$  und  $A'$ .

**Produktautomat:**  $A_{\times} := (Q \times Q', \Sigma, \delta_{\times}, (s, s'), F_{\times})$  mit

$$\delta_{\times}((q, q'), a) = (\delta(q, a), \delta(q', a))$$

Definiere  $F$  je nach Mengenoperation:

$$L \cup L': F_{\times} := Q \times F' \cup F \times Q'$$

$$L \cap L': F_{\times} := F \times F'$$

...



# Einfache Eigenschaften endlicher Automaten

$$L(A) = \emptyset?$$

Leerheit

$\Leftrightarrow \neg \exists f \in F : f$  ist von  $s$  aus **erreichbar**

$\rightsquigarrow$  Tiefensuche, **Linearzeit**, auch für **NEA**.

$$L(A) = \Sigma^*?$$

Vollständigkeit

$\Leftrightarrow \neg \exists q \in Q \setminus F : q$  ist von  $s$  aus **erreichbar**?

$\rightsquigarrow$  Tiefensuche, **Linearzeit**, nur für **DEA**!

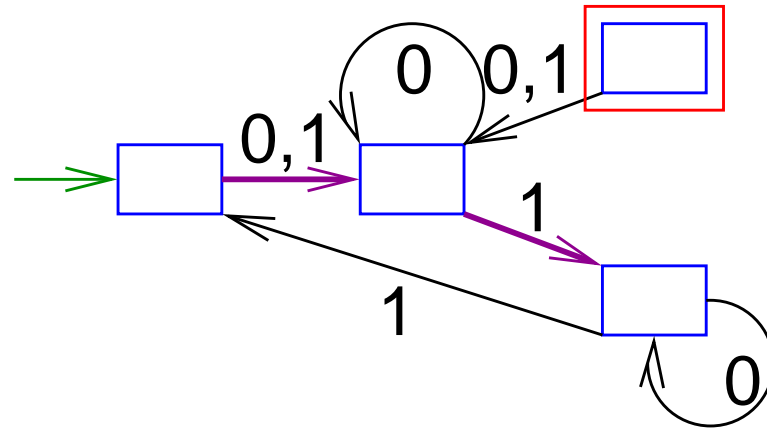
Vollständigkeit für NEA:

Umwandlung in DEA. Nichts deutlich besseres bekannt.



# Beispiel

$$L(A) = \emptyset$$



Kante im  
Tiefensuchbaum



# Unendlichkeit

$|L(A)| = \infty? \Leftrightarrow \exists$  akzeptierender Pfad, der Kreis enthält.

OBdA: NEA mit  $F = \{f\}$ . Sei  $G_A = (Q, E)$ ,

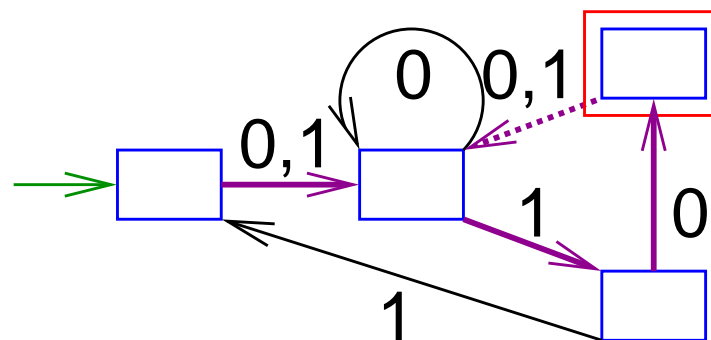
$$E = \{(q, r) : \exists a \in \Sigma \cup \{\varepsilon\} : r \in \delta(q, a)\}$$

1. Entferne Zustände, von denen aus  $f$  nicht erreichbar ist.

Tiefensuche in  $\bar{G}_a = (Q, \{(q, r) : (r, q) \in E\})$  von  $f$ .

2. Ist von  $s$  aus ein Kreis erreichbar?  $\Leftrightarrow$

Trifft Tiefensuche in  $G_A$  von  $s$  auf eine Rückwärtskante?



Kante im  
Tiefensuchbaum

Rückwärtskante im  
Tiefensuchbaum



# Äquivalenz von DEAs

$L$  und  $L'$  seien reguläre Sprachen definiert durch DEAs  $A, A'$ .

Frage  $L = L'$ ?

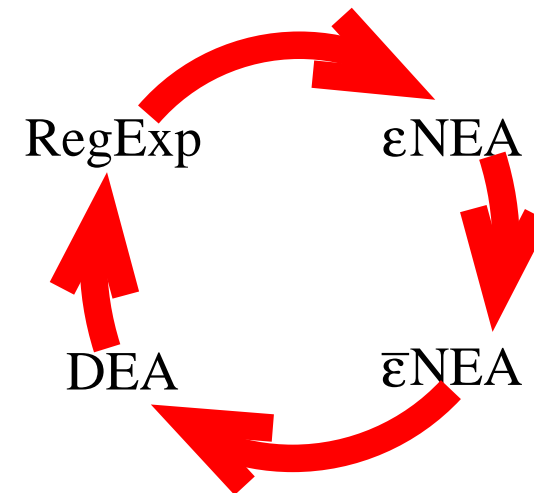
$$\Leftrightarrow \neg \exists w : (w \in L \wedge w \notin L') \vee (w \notin L \wedge w \in L')$$

$$\Leftrightarrow \neg \exists w : (w \in L \wedge w \in \bar{L}') \vee (w \in \bar{L} \wedge w \in L')$$

$$\Leftrightarrow (L \cap \bar{L}') \cup (\bar{L} \cap L') = \emptyset$$

z.B. via Produktautomat

Problem: langsam





## Äquivalenz von DEAs

$L$  und  $L'$  seien reguläre Sprachen definiert durch DEAs

$$A = (Q, \Sigma, \delta, s, F), A' = (Q', \Sigma, \delta', s', F').$$

Idee: der **Minimalautomat** ist „eindeutig“.

$\rightsquigarrow$  minimiere beide Automaten und prüfe auf „Gleichheit“.

Problem: Zustandsumbenennungen sind erlaubt. Die Komplexität des **Isomorphietestes** allgemeiner Graphen ist ein offenes Problem.



# Äquivalenz von DEAs

$L$  und  $L'$  seien reguläre Sprachen definiert durch DEAs

$A = (Q, \Sigma, \delta, s, F)$ ,  $A' = (Q', \Sigma, \delta', s', F')$ . ObdA  $Q \cap Q' = \emptyset$ .

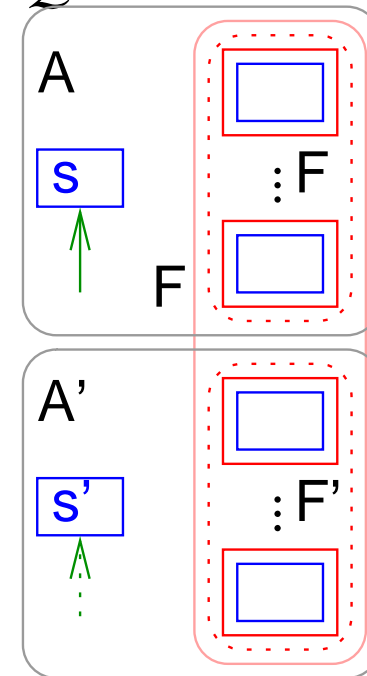
Frage  $L = L'$ ?

Betrachte  $A_{\cup} := (Q \cup Q', \Sigma, \delta_{\cup}, s, F \cup F')$  mit

$$\delta_{\cup}(q, a) = \begin{cases} \delta(q, a) & \text{falls } q \in Q \\ \delta'(q, a) & \text{falls } q \in Q' \end{cases}$$

Bestimme Äquivalenzklassen der Zustände von  $A_{\cup}$ .

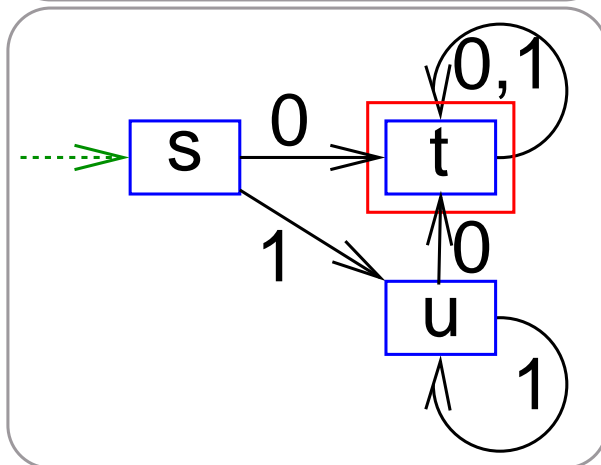
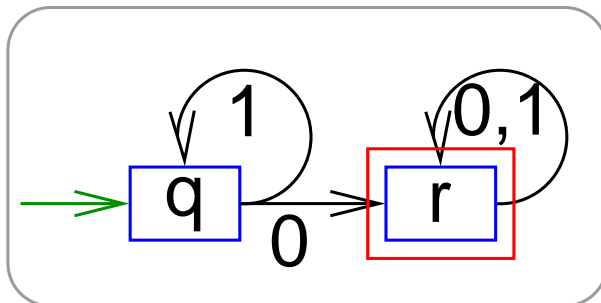
$$L = L' \Leftrightarrow s \equiv s'.$$





# Beispiel

$L \subseteq \{0, 1\}^*$  Sprache aller Wörter mit mindestens einer Null



Algorithmus zur Markierung aller inäquivalenter Zustandspaare liefert:

$\{q, r\}, \{q, t\}, \{s, r\}, \{s, t\}, \{u, r\}, \{u, t\}$

$\rightsquigarrow q \equiv s$

$\rightsquigarrow$  Beide Automaten sind äquivalent.



# Zusammenfassung Endliche Automaten u. Reguläre Sprachen

- Einfaches Maschinenmodell
- Umfassend algorithmisch beherrschbar (Zustandsminimierung, Konvertierung mit regulären Ausdrücken, . . .)
- Akzeptierte Sprachen vollständig verstanden
- Nützliche Anwendungen: Textverarbeitung, Compiler, Hardware, . . .
- Konzept des Nichtdeterminismus